# Ethernet-Controller

## 3C200

**-Reference Manual-**

D923408e

Document-No. D923408e

7/27/82

This is an addendum to the

Model 3C200

Q-Bus Ethernet (QE) Controller

Reference Manual

of

December 15, 1981

CORRECTIONS:

Three jumpers have been added to the QE Control Board. These jumpers allow the user to selectively disable a portion of the QE buffer memory. Disabled buffers do not respond to any bus requests. Therefore, normal Q-Bus memory may occupy that space, or a gap may be left in the address space between the top of normal memory and the bottom of QE memory. The jumpers are as follows:

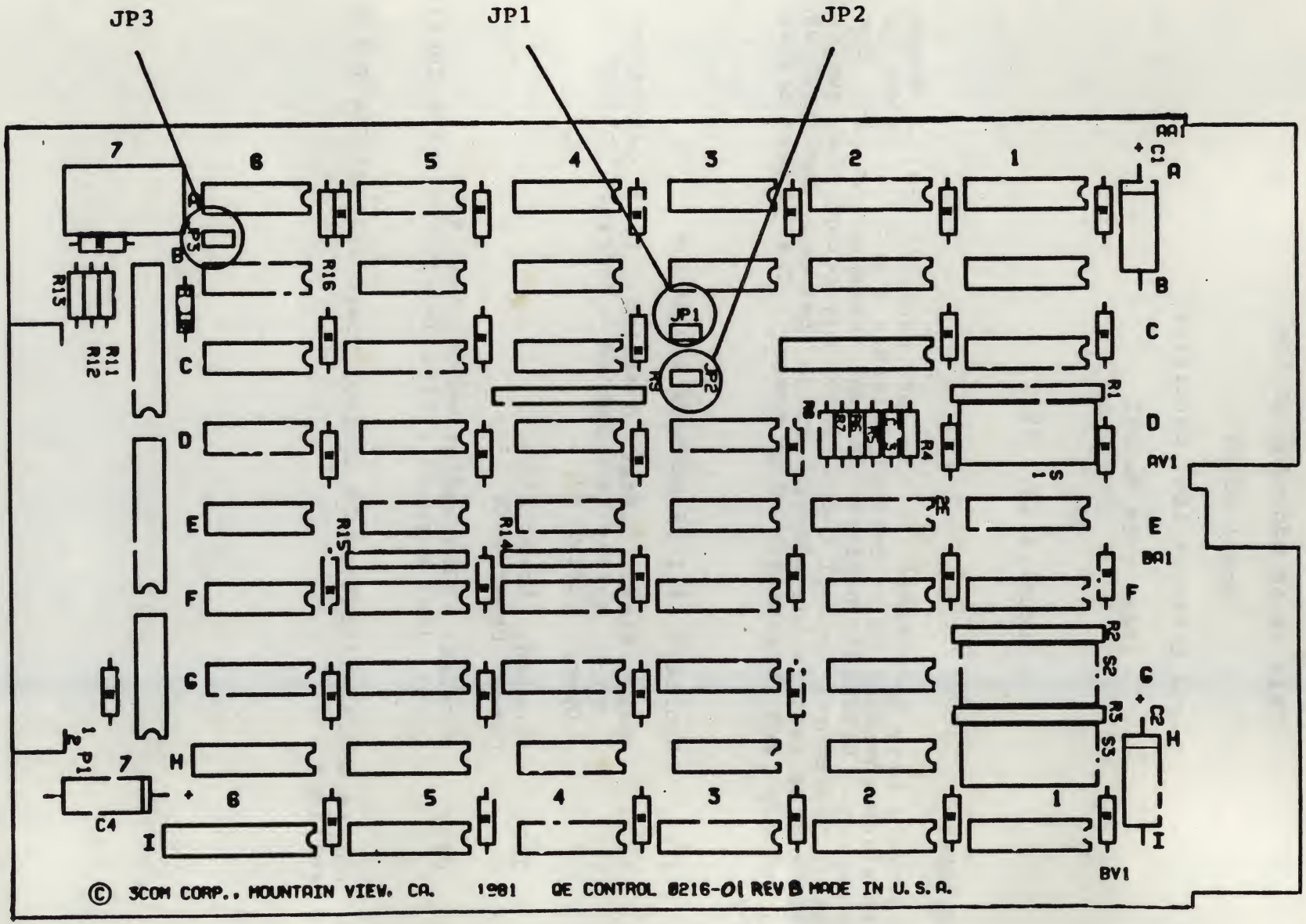
JP1: Shorted: All 16 buffers available

Open: First buffer disabled. (JP2 determines which buffer is disabled - Shorted: buffer 0, Open: buffer 8)

JP2: Shorted: Lower 16KB
Open: Upper 16KB

JP3: Shorted: 32KB buffer memory available
Open: 16KB available (JP2 determines which 16KB)

These jumpers are available on QE Control Board 0216-01 Rev B and later revisions.


JP3
JP1
JP2
7
6
5
4
3
2
1
A
B
C
D
E
F
G
H
I
R13
R11
R12
R16
R15
R14
R1
R2
R3
R4
C1
C3
C4
P1
AV1
BA1
BV1
© 3COM CORP., MOUNTAIN VIEW, CA. 1981 DE CONTROL 8216-01 REV B MADE IN U.S.A.

3Com Corporation
computer communication compatibility

# TABLE OF CONTENTS

## CHAPTER 8 - RSX-11M DRIVER



## APPENDICES

# CHAPTER 1

# 3COM QE CONTROLLER

# SPECIFICATIONS

## 1.1 DESCRIPTION

The 3C200 Qbus Ethernet Controller provides the connection to Ethernet for LSI-11 family computers. It consists of a set of three double-height (dual) modules that plug into the Qbus (LSI-11 bus) used on the LSI-11, LSI-11/2, LSI-11/23, PDP-11/03, and PDP-11/23. See Figure

1-1 below for photo of board set.

Connection to Ethernet is made via the 3Com Ethernet transceiver and cable or any other Ethernet compatible transceiver and cable.

The controller is built around a 32K byte dual-ported memory. One memory port is used for packet transfers between buffer memory and Ethernet, without consuming Qbus or LSI-11 cycles. The other memory port allows the LSI-11 to directly access the full 32K bytes to process packets.

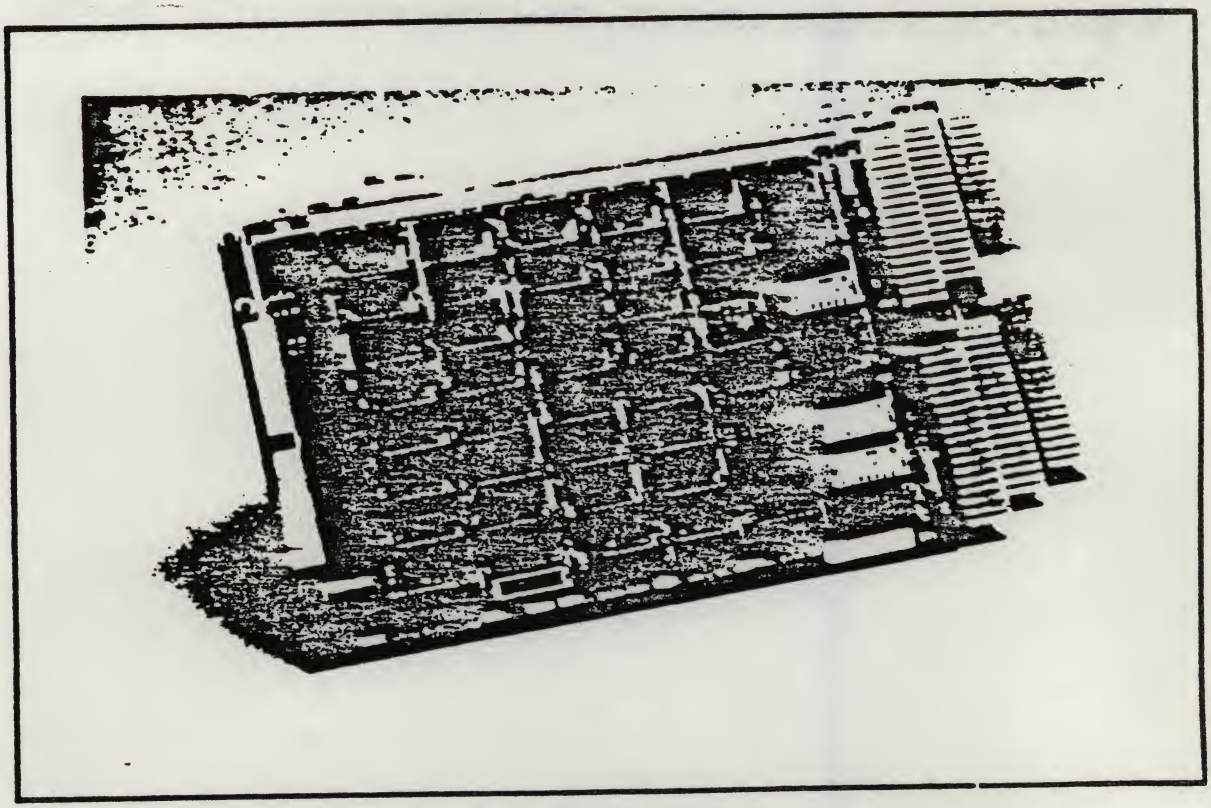


FIGURE 1-1. 3COM Q-BUS ETHERNET CONTROLLER BOARD SET

Under software control, a portion of this memory is allocated to transmit and receive buffers of 2K bytes each, with the remainder available for ordinary program and data storage. Since many Ethernet applications require no more than four buffers of 2K bytes each, the system memory is effectively increased by 24K bytes.

The 3C200 Controller, 3C100 Ethernet Transceiver and the 3C110 Transceiver Cable conform to the Ethernet specifications, version 1.0, published by DEC, Intel, and Xerox on 30 September, 1980. When coupled with any 3Com software driver, they implement layers one (physical) and two (data link) of the International Standards Organization Reference Model for Open Systems Interconnection. Any LSI-11 family computer so equipped will be compatible with any other Ethernet-based system at the physical and data link levels.

Software drivers are available for UNET [1], 3Com's communications software, for the networking of UNIX[2]-based systems and for DEC's RT-11 and RSX-11M operating systems. A system using UNIX/UNET provides communication through all seven layers of the ISO model, with UNET providing file transfer, virtual terminal, electronic mail and process-to-process communication.

1 UNET is a trademark of 3Com Corporation.
2 UNIX is a trademark of Bell Laboratories.

## 1.2 QE FEATURES

- Compatible with 10 megabit-per-second DEC, Intel, Xerox Ethernet.

- Compatible with DEC Qbus (LSI-11 bus) allowing use with LSI-11/2, LSI-11/23, PDP-11/03, and PDP-11/23.

- Connects to Ethernet using 3Com Ethernet Transceiver and Transceiver Cable or any other transceiver and cable that conform to the Ethernet specification.

- Controller handles serial-parallel and parallel-serial conversions, buffering for transmission and reception, framing of packets, encoding and decoding, collision and error detection.

- Includes 32K byte dual-ported memory that appears in the LSI-11 address space. Transfers between the dual-ported memory and Ethernet do not consume Qbus or LSI-11 cycles, allowing concurrent processing.

- 2K byte buffers can handle maximum packet size allowed by Ethernet specification with up to 16 transmit/receive buffers queued, controlled by independent FIFOs.

- Partitioning of dual-ported memory is under software control for optimal utilization.

- Can transmit and receive minimally spaced packets.

- Manchester decoding using phase-locked loop circuitry.

- 32-bit CRC generated on transmit and verified on receipt to ensure data integrity.

- Contained on three DEC-standard double-height modules compatible with LSI-11 chassis.

- UNET users are provided communication through all seven layers of ISO model.

- RT-11 and RSX-11M drivers provide communication through physical and data link layers of the ISO model.

(THIS PAGE BLANK)

## 1.3 QE SPECIFICATIONS

### Compatibility

| | |
|---|---|
| Ethernet | Conforms fully to Ethernet specification, version 1.0, published 30 September, 1980, by DEC, Intel, and Xerox. |
| Qbus | Conforms fully to Qbus (LSI-11 bus) as specified by DEC. |

### Functions

Serial/parallel and parallel/serial conversions.
Transmit and receive buffering.
Framing of packets.
Manchester encoding and decoding.
Collision and error detection.

### Buffer memory

| | |
|---|---|
| Size | 32K bytes. |
| Type | Dual-ported RAM memory. |

| | |
|---|---|
| Cycle time | 400nsec. |
| Refresh | Every 1.64msec. |
| Buffer size | Transmit - 2K bytes each.<br>Receive - 2K bytes each. |
| Organization | 1-16 buffers allocated under software control, remainder used by LSI-11 for program or data storage. |
| Byte ordering | Low order first or high order first, jumper selectable. |
| Buffer control | Independent transmit and receive FIFO register queues (16 x 4). |
| Address | Occupies 32K bytes of LSI-11 memory with starting address set by switches at any 32K byte address boundary. Bus addressing of 18 or 22 bits is switch selectable. |

**Ethernet Packet Format**

| | |
|---|---|
| Length | 512 bits minimum, 12,144 bits maximum, excluding 64 bit preamble generated and deleted by hardware. |
| Format | Destination Address.....48 bits<br>Source Address.....48 bits<br>Type.....16 bits<br>Data.....8n bits<br>where n=46 minimum, 1500 maximum<br>Frame Check Sequence.....32 bits |
| Frame check sequence | 32 bit CRC generated on transmit, verified on receipt. |

**Timing**

| | |
|---|---|
| Bit rate | 10 million bits per second. |
| Transmit turnaround | 10.2 microseconds. |
| Receive turnaround | 9.6 microseconds. |
| Jam time | Transmits 32 bits of zeros when collision |

is detected.

## Interrupts

| | | |
|---|---|---|
| Interrupt vector address | Receive completed | 0000\|000x\|xxxx\|0000 |
| | Transmit jam/collision | 0000\|000x\|xxxx\|0100 |
| | Transmit completed | 0000\|000x\|xxxx\|1000 |
| | Bits 4-8 are switch selectable | |

| | | |
|---|---|---|
| Priority levels | Receive completed | 6 |
| | Transmit jam/collision | 5 |
| | Transmit completed | 4 |

## I/O Register Addresses

| | |
|---|---|
| Receive control | 111x\|xxxx\|xxxx\|xx00 |
| Transmit control | 111x\|xxxx\|xxxx\|xx10 |
| Bits 2-12 are switch selectable | |

## Software Functions

The following functions are performed by software:

Address recognition.
Retransmission timing after collision.
Runt filtering, the elimination of received packets below minimum size.

## Installation

| | |
|---|---|
| Size | Three DEC-standard double modules, each 13.2cm x 23.4cm (5.2in x 8.5in) joined by a ribbon cable. |
| Slots | Requires three adjacent double module slots. |
| Power | 3.4A typical at +5V nominal<br>0.8A typical at +12V nominal |
| Bus loading | AC - 2 DEC unit loads<br>DC - 2 DEC unit loads |
| Memory | Memory management unit (MMU) required when total memory, including the 32K byte controller memory, exceeds 64K bytes. |
| Transceiver cable connector | Ethernet-standard female 15-pin "D" subminiature connector attached to the controller via ribbon cable. Optional LSI-11 chassis mount included. |
| Transceiver cable | Uses Ethernet-standard transceiver cable |

which must be ordered separately.

| | |
|---|---|
| Ethernet address | Unique address supplied by 3Com for each controller. |

**Operating Environment**

| | |
|---|---|
| Temperature | 5° to 55° C |
| Humidity | 10% to 90% without condensation |

**Software**

3Com UNET support

Driver for RT-11

Driver for RSX-11M

Standalone diagnostic

# CHAPTER 2

## BACKGROUND

This chapter covers some backround information about both the Ethernet and the 3Com QE Controller. Readers who are already familiar with Ethernet may wish to skip to Chapter 3.

In the last decade, computers have grown from a luxury to a necessity in most businesses. Similarly, in the next decade, inter-computer communication will grow from a luxury to a necessity.

The "Ethernet" network, developed for machine-machine communication, was pioneered at Xerox Corporation as an appropriate implementation for inter-computer communications. In use since 1974, the Ethernet has evolved to an industry standard, documented in the Ethernet Specification, published September 30, 1981 by DEC, Intel, and Xerox.

The benefits of modern computerized workstations are now magnified when they can communicate information to other devices at 10 million bits-per-second over the Ethernet. What's more the Ethernet network can be tailored to end user's needs and workstations once the Ethernet coaxial cable is in place. This means multiple workstations can share the same resources such as:

| | |
|---|---|
| Word processors | Data bases |
| Printers | Process control stations |
| Electronic mail systems | Array processors |
| Graphics stations | Laser printers |
| Transaction workstations | Etc. |

Individual workstations and other devices are plugged into Ethernet information outlets in the wall the same way telephones are plugged into telephone wall-outlets. However, the Ethernet has a more complex connection with 15-pins.

Each Ethernet device is assigned a unique address (like a unique telephone number), therefore, devices can be moved around and plugged into any convenient Ethernet information outlet. Further, all devices plugged into the Ethernet can talk to each other by mutual agreement similar to two people talking on the telephone.

Ethernet, due to its standardized physical and logical protocol, allows users to mix and match equipment from multiple vendors.

In the future a voice capability will probably be integrated into Ethernet for store-and-forward voice communications to complement electronic mail.

## 2.1 BASIC ETHERNET SUBSYSTEMS

The Ethernet is a bus-oriented communication system that supports up to 100 stations using a 50 ohm coaxial cable as the bus.

**Figure 2-1** below shows the basic parts of a typical Ethernet system, with workstations connected to the Ethernet coaxial cable.

The **Transmission Subsystem** is made up of 50 ohm coaxial cable, terminators, transceivers, and transceiver cables.

The **Controller Subsystem** is the set of controller

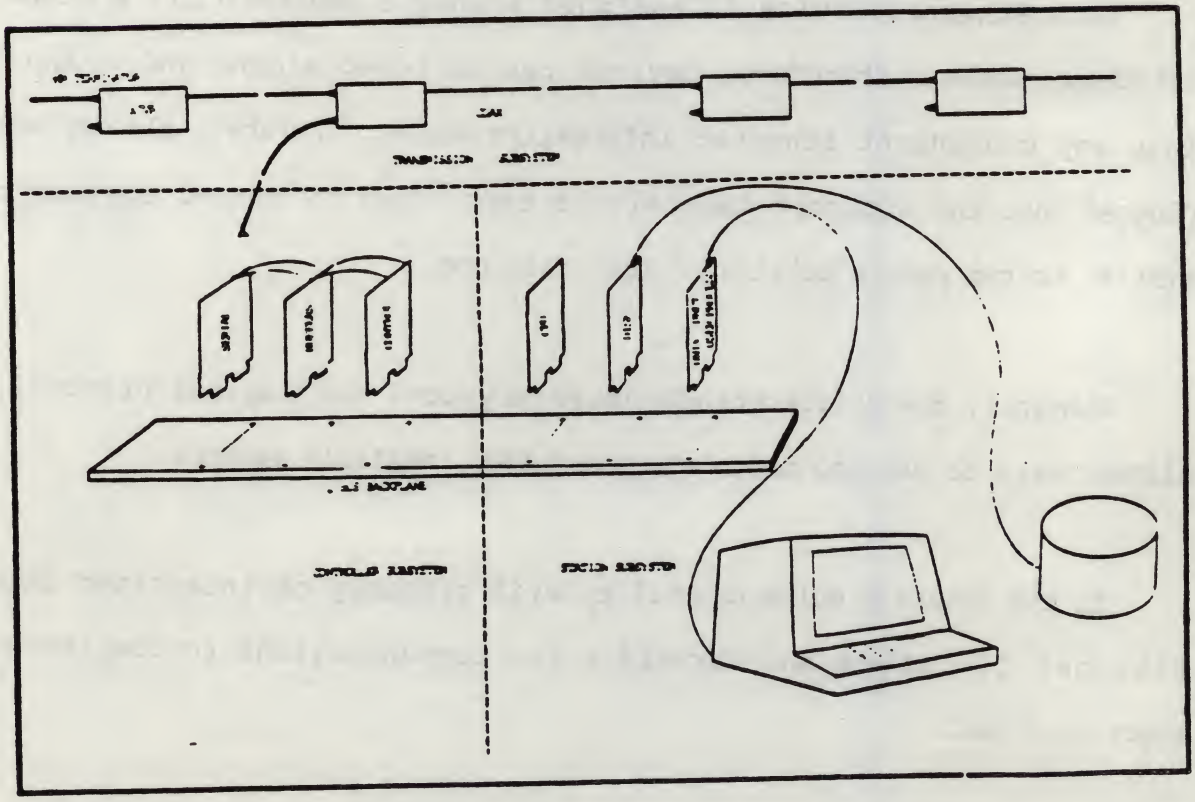


**FIGURE 2-1. BASIC ETHERNET SUBSYSTEMS**

boards and the software supporting them.

The **Station Subsystem** is everything else associated with the station, such as, the terminal, processor, disk, and higher level protocol software.

These three subsystems are discussed again later in this chapter, in terms of 3Com product implementation.

Meanwhile, the following example describes how a file of information is transferred from one device to another using Ethernet.

## 2.2 EXAMPLE FILE TRANSFER

(This is an example text file transfer using a File Transfer Program running on the host processor.)

1. The terminal user runs the File Transfer Program, connects to the receiver, and specifies the file to be transferred.

2. The file's characters are mapped into device-independent virtual characters (by software) to meet protocol specifications.

3. The mapped character stream is then routed to a virtual circuit set up between the two devices.

4. The virtual circuit protocol software breaks the character stream into packets for transmission. (It also retransmits corrupted packets, limits data rate to avoid overruns, and multiplexes other virtual circuits.)

5. The packets are then passed to the Ethernet driver software.

6. The Ethernet driver then copies the packet into a packet buffer and tells the controller to transmit it.

7. The controller waits until the coaxial cable is not in use, then transmits the packet.

8. The Ethernet transceiver receives the packet's bit stream and injects it onto the coaxial cable. (If the transceiver detects a collision, it signals the controller to retransmit.)

9. The receiving station recognizes its address and

reverses the above procedure: bits are received by the transceiver, fed to the controller, passed to software that reassembles the packets, maps the characters, and stores the data.

## 2.3 HOW 3COM PRODUCTS IMPLEMENT ETHERNET FOR DEC COMPUTERS

For a complete local computer network, there are only three additional components needed by systems running the UNIX operating system on DEC VAX-11, PDP-11, or LSI-11 computers:

1. **3Com Ethernet Transceiver** - fully conforms to published Ethernet specifications and connects directly to the Ethernet coaxial cable.

2. **3Com Ethernet Controller** - available in two models that plug directly into either the LSI-11 or PDP-11/VAX-11 computers.

3. **3Com Higher-level Protocol Software** - for example the UNET Software package that provides high-level network protocol services for UNIX - including data link drivers, the Internet Protocol (IP), Transmission Control Protocol (TCP), file transfer protocol (UFTP), electronic mail protocol (UMTP), virtual terminal protocol

(UVTP), etc.

## 2.4 ETHERNET OPERATION

The Ethernet is a carrier sense, multiple access transmission system with collision detection (CSMA/CD). To transmit a packet, a station waits for quiet on the network (defers). When the network is quiet, it starts to transmit the packet.

During packet transmission, the station also watches for collisions with other transmitters; these may occur within one round-trip time through the network. The station is said to have "acquired the network" if no collision occurs in that time interval. If a collision does occur, the station transmits 4 to 6 additional bytes of data (jam) and the aborts the packet. The extra bytes insure that any other participant in the collision is sure to see it. The station then waits a random amount of time (backoff) before retransmitting (after deferring to packets in progress on the network).

## 2.5 TRANSMISSION SUBSYSTEM

The transmission subsystem in the form of a 3Com "starter package",

(Model 3C140) is shown in Figure 2-2 below. It consists of four types of components: transceiver cables, transceivers, coaxial cable, and terminators. These are described below.

**Transceiver Cable** - The transceiver cable is a 15 meter shielded twisted pair cable that connects the controller to the transceiver. It has 4 pairs, one each for transmit, receive, collision detect, and power. It has a male 15 pin connector with lock posts on the controller end and a female 15 pin D connector with slide lock assembly on the transceiver end. Thus the cables can be concatenated to make a longer cable, up to the maximum length of 50 meters.

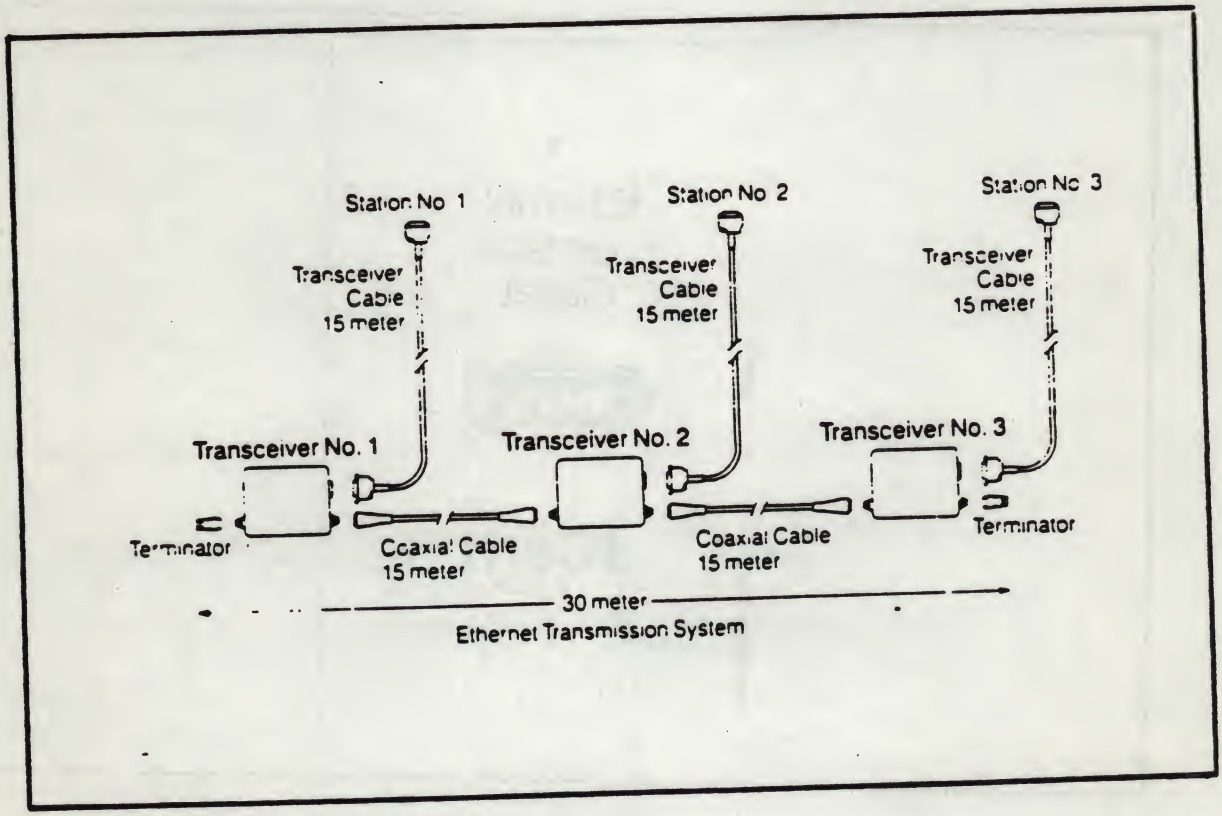



FIGURE 2-2. 3COM ETHERNET TRANSCEIVER STARTER PACKAGE

To minimize EMI, the connectors have internal shields connecting the cable shield to the shell of the connector. The male cable connector can either be brought out of the wall to the station or mounted on a cover plate providing a bulkhead disconnect at the wall. When mounted on the cover plate, it has been referred to as the "Information Outlet." (see Figure 2-3 below)

**Transceiver** - The 3Com transceiver is compatible with the DEC, Intel and Xerox Ethernet specification.

The transceiver makes a high impedence connection to the common coaxial cable and provides electrical isolation between the coaxial cable

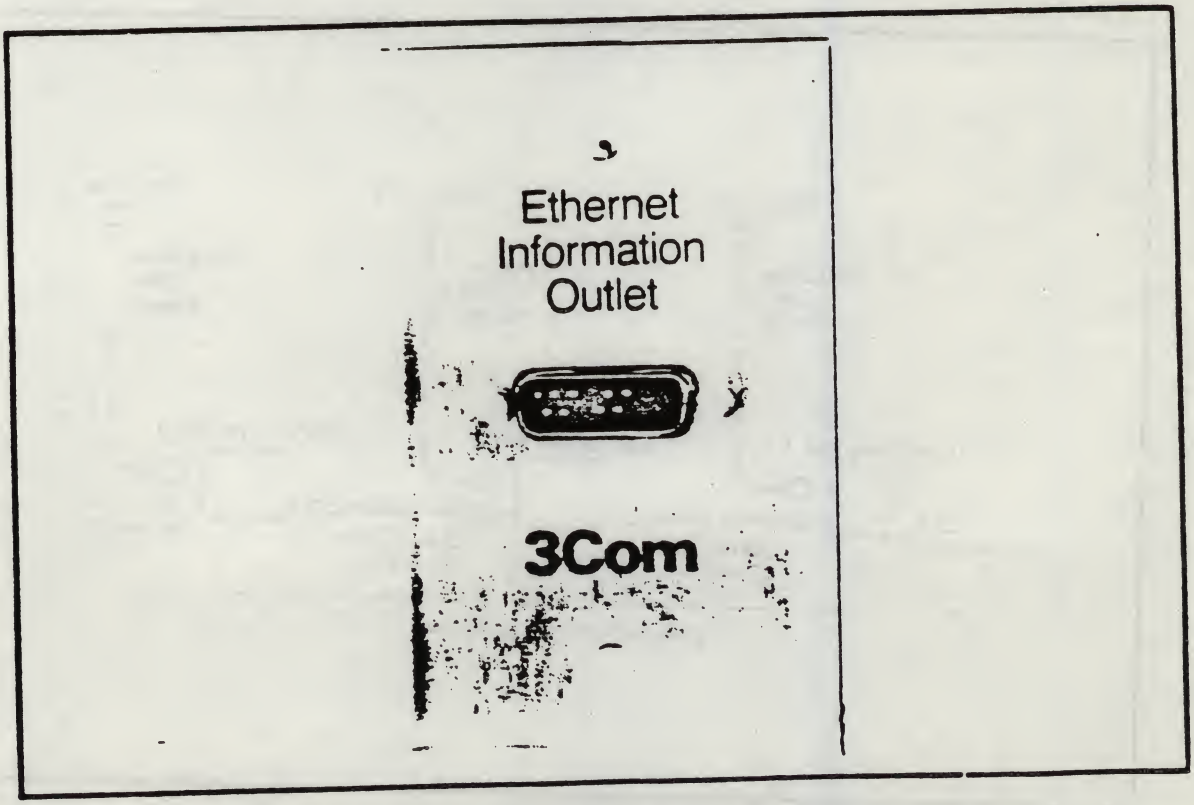



FIGURE 2 - 3. INFORMATION OUTLET

and the twisted pair cable.

Transmit signals from the controller are injected into the coaxial cable by the transceiver. Signals from the coaxial cable appear on the receive lead of the transceiver cable with balanced signalling.

The receiver also provides correction for signal distortions caused by traveling through long lengths of coaxial cable.

The collision signal appears if there is a signal present from any other station on the network. When transmitting this indicates a collision. When not transmitting, it indicates the presence of other signals on the network.

**Coaxial Cable** - The coaxial cable is a 50 ohm cable with multiple shields to minimize susceptability to strong RF fields.

**Cable Connectors** - Cable sections are terminated with standard N-series connectors. Rubber boots cover the connectors to prevent inadvertent connection of the coaxial shield to building grounds. Multiple ground connections of the coaxial shield to building grounds are a potential source of ground induced noise into the coaxial shield. Coaxial cable sections are joined by insulated barrel connectors (N-Series female-female adapters).

**Terminators** - The ends of a coaxial cable segment are terminated with

50 ohm terminators with insulated outside covers.

## 2.6 QE CONTROLLER SUBSYSTEM

The QE Ethernet Controller interfaces the transceiver to the internal bus of the computer to which it is connected. It performs serial-parallel and parallel-serial conversion, buffering, CRC generation and checking, address recognition, phase encoding and decoding. The I/O structure and speed of the processor determine how these functions are partitioned between hardware and software (or microcode) in the Ethernet station.

**Buffering:** The amount of buffering needed in the controller is determined by the latency and speed of the Ethernet channel. Most processors have bus transfer rates that are unduly stressed by the 10Mbps Ethernet bandwidth, therefore, full packet buffers are necessary in order to keep up with the bit rate of network traffic.

**CRC Generation And Checking:** The cyclic redundancy code (CRC) uses the 32 bit polynomial from the U.S. Department of Defense Autodin II system. The choice of where to implement this function is based on expected traffic load for a given machine and available processor resource.

**Address Recognition:** The controller watches every packet that passes to determine whether to accept the packet, based on its destination

address. The destination address must be checked quickly to avoid processing unwanted packets. The QE Controllers 16 full-size packet buffers reduce the possibility of dropped packets should the processor fall behind while checking addresses. Software address recognition minimizes the hardware content of the controller.

**Phase Encoding, Decoding, and Transceiver Interface;** Manchester encoding is used for data transmission on the Ethernet. It has a 50% duty cycle. The first half of a bit cell contains the complement of the bit and the second half of the bit cell contains the bit.

**Phase Encoding** is done in the controller by exclusive-ORing the clock with the data. (Decoding is also performed in the controller. Partitioning of encode-decode functions into the controller rather than the transceiver minimizes wires to the transceiver while minimizing transceiver size and power dissipation.)

**Phase Decoding** in the QE Controller is done by an analog phase-locked loop technique. This technique has the advantage of tolerating more phase jitter than alternative techniques in use, twice the tolerance of a typical one-shot decoder and four times the tolerance of a typical digital state-machine decoder.

**The Transceiver Interface** contains line drivers and receivers.

# CHAPTER 3

# PHYSICAL DESCRIPTION

## 3.1 ENVIRONMENT

The QE Controller interfaces a Digital Equipment Corporation LSI-11 microcomputer to an Ethernet transceiver.

The QE Controller plugs into the backplane of the LSI-11 and resides in the same enclosure with it. An Ethernet transceiver cable, of approximately 50 feet long connects the QE Controller to the Ethernet transceiver. The Ethernet transceiver in turn taps directly into the Ethernet coaxial cable.

According to the International Standards Organization Open Systems Interconnection Reference Model, the QE Controller performs a link layer

service, the second of seven layers of service (see Figure 3-1 below).

## 3.2 PACKAGING

The Qbus Ethernet Controller (QE) gives the LSI-11 access to the Ethernet. Together with an Ethernet transceiver, it can transmit and receive packets through the common coaxial cable. Physically, the controller consists of three dual height cards (each 5.2" x 8.5") that

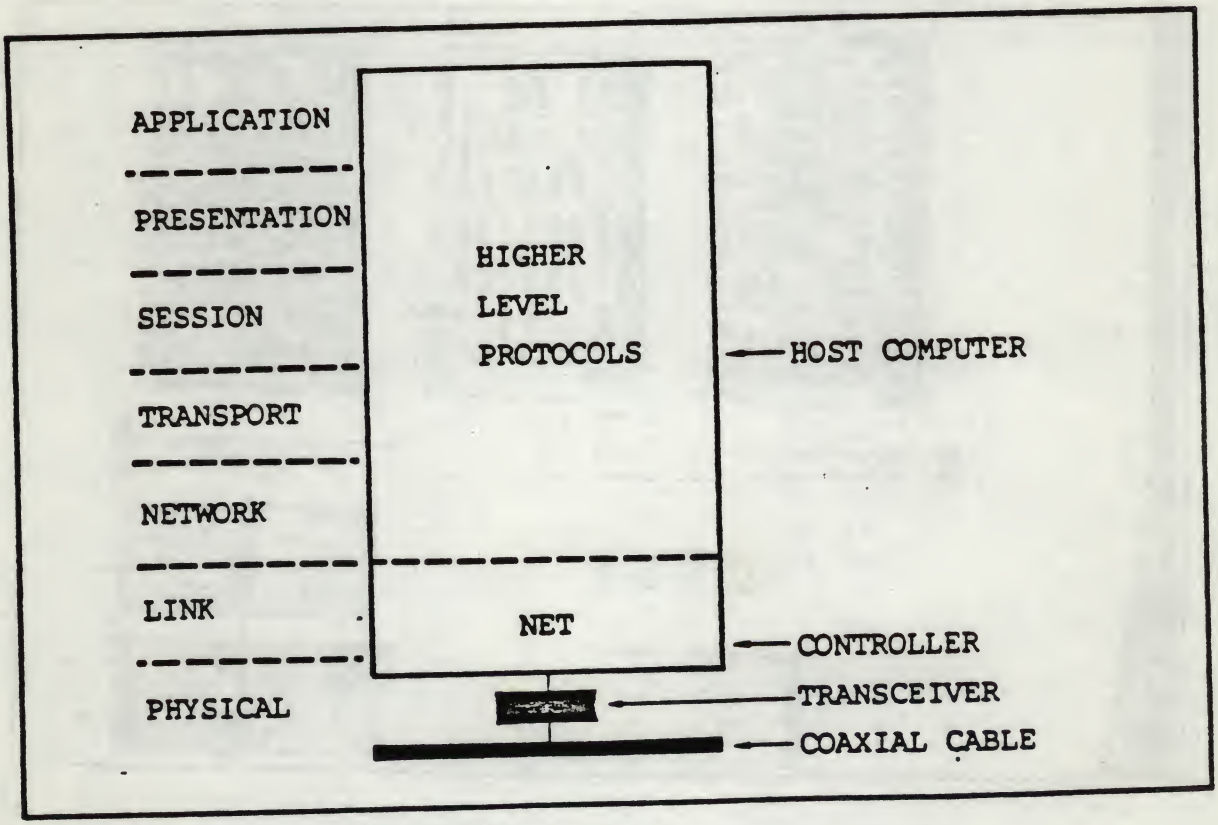



FIGURE 3-1. ISO PROTOCOL HIERARCHY

plug into a standard Qbus backplane as shown in Figure 3-2 below.

The three DEC dual-sized PC boards are:

**Memory Board**

**Control Board**

**Serial Board**

The three boards are interconnected by a short ribbon cable. The 50-pin connectors for the jumper cable are located on the edge opposite the fingers that plug in to the LSI-11 backplane. The serial board has an additional connector (next to the interboard jumper connector) for a

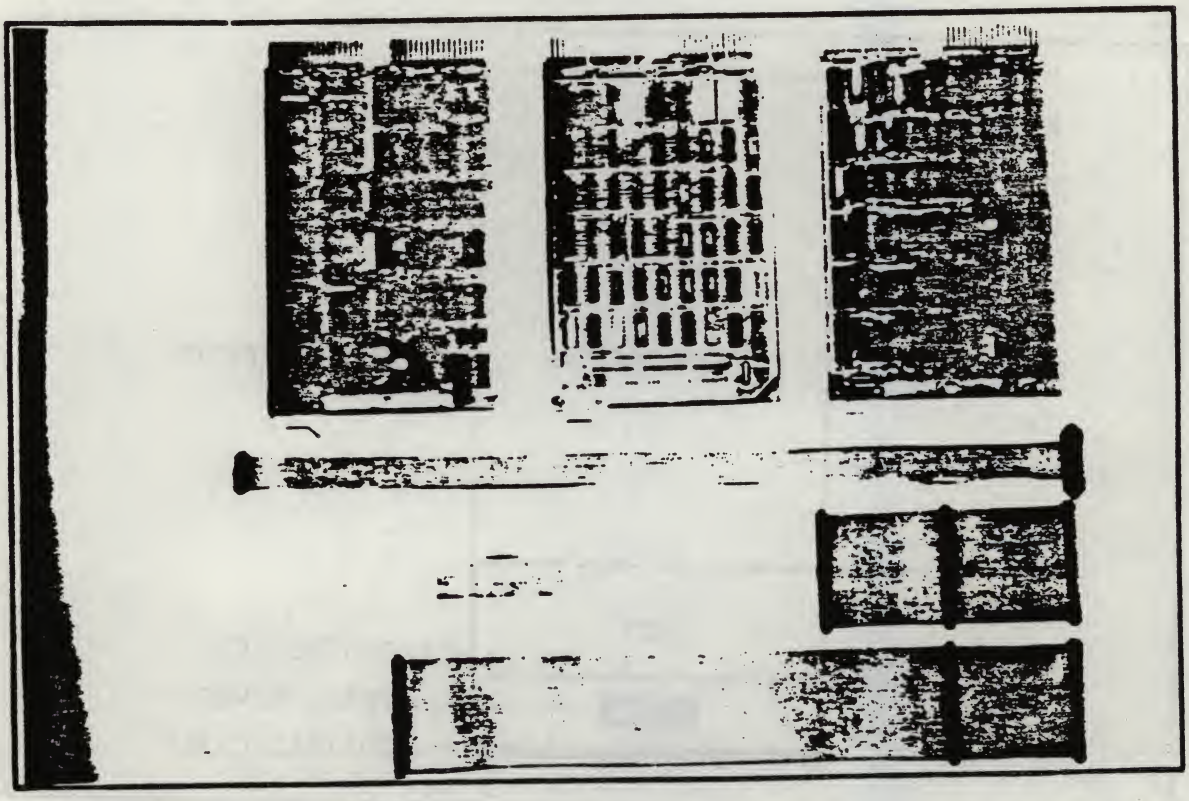


FIGURE 3-2. QE PACKAGING ENVIRONMENT

short cable that mates with the transceiver cable. This connector may be fastened to the metal LSI-11 chassis using a metal L-bracket (included) to reduce Electromagnetic Interference (EMI).

Two varieties of the interconnect cable are available. The standard variety is designed to interconnect the three boards when configured in adjacent slots, like a stack of pancakes. The longer "L" variety allows one of the three boards to be moved into an adjacent slot in the same plane as one of the other two boards, providing greater configuration flexibility. (See Chapter 5)

The QE memory board contains the 32K byte dual-ported memory for the sixteen 2K transmit and receive buffers. One memory port makes 32K directly accessible to the LSI-11 processor via the Qbus. The other memory port passes data to/from the serial board.

The QE control board implements the following:

- Interrupt and timing logic
- Four 16 by 4-bit FIFO memories for the following buffer numbers: buffers waiting to be transmitted, buffers finished transmitting, buffers available for receiving, and buffers filled with incoming packets.

The QE serial board implements the following:

- Receive and transmit state machines that control the framing of packets
- Shift registers for parallel to serial conversion and vice versa
- CRC calculation and verification logic
- Receiver synchronization logic

The QE Controller can be used with any transceiver that conforms to the DEC-Intel-Xerox Ethernet specifications.

### 3.3 BLOCK DIAGRAM

The QE Controller features visible to the LSI-11 via the Qbus are:

(See Figure 3-3).

- 32K-byte memory
- 16-bit transmit control and status (register) (QEXCR)
- 16-bit receive control and status (register) (QERCR)

Data packets for transmission are placed into the 32K buffer memory by the LSI-11 processor. From there, directed by the control logic, successive 8-bit bytes are fetched from the buffer memory and placed into a shift register. The serialized data is fed simultaneously to both the

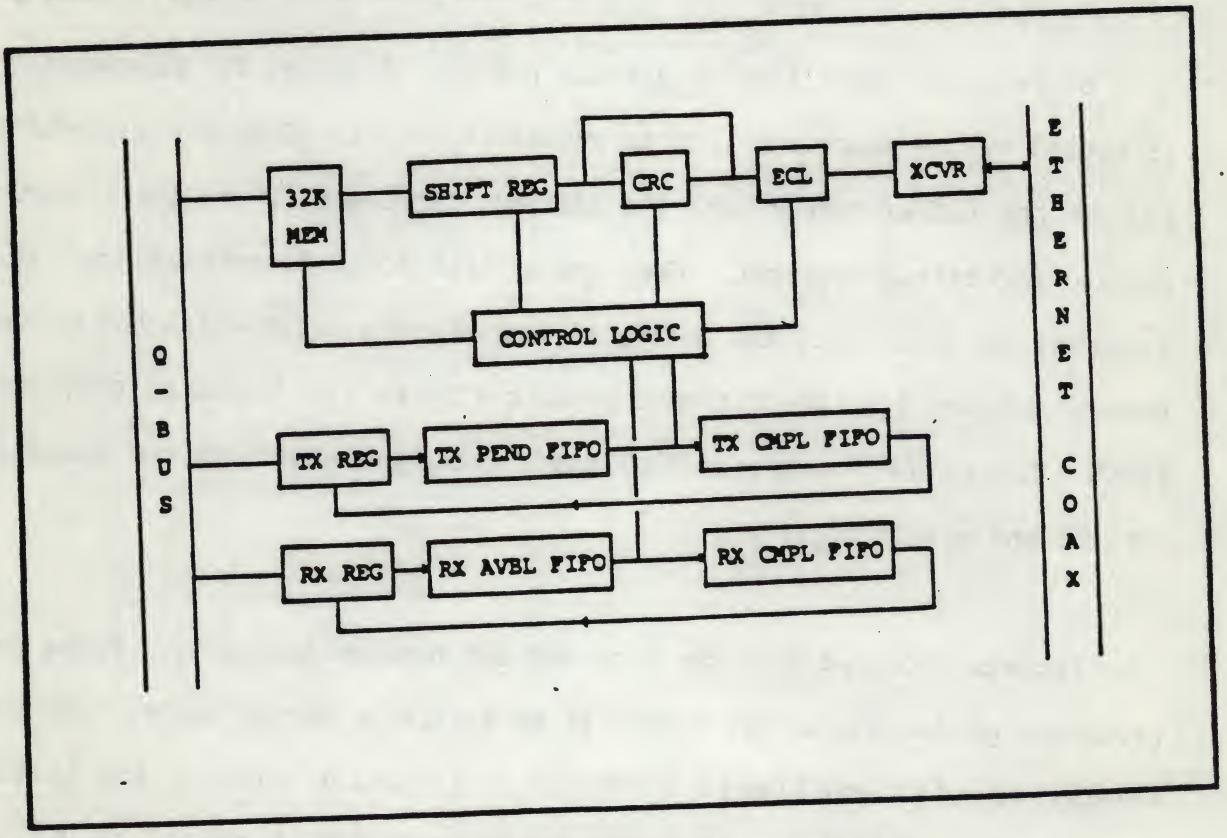



FIGURE 3-3. QE CONTROLLER FUNCTIONAL BLOCK DIAGRAM

CRC (Cyclic Redundancy Check) logic, and the ECL front end logic connected through the transceiver cable to the Ethernet coaxial cable.

Received data packets follow exactly the reverse path. Data on the coaxial cable flows through the Ethernet transceiver to the receiver synchronizer portion of the ECL front end logic. The data then travels simultaneously to both the CRC logic and to the shift register where 8-bit bytes are assembled for storing into the 32K buffer memory.

The 32K buffer memory is treated as 16 2K-byte buffers. Each 2K buffer can hold one transmit packet or one receive packet, or it can be used like regular memory on the Qbus for storing program code or data.

Buffers are identified by a 4-bit number. A packet for transmission is stored by the processor into an available buffer. Then the processor places the buffer number into the transmit pending FIFO via the transmit control and status register. When the control logic determines that the Ethernet is quiescent, the packet is transmitted, after which the buffer number is moved from the transmit pending FIFO to the transmit complete FIFO. The buffer number is then read by the processor via the transmit control and status register.

Packets received from the Ethernet are handled similarly. First the processor places the buffer number of an available packet buffer into the receive buffer available FIFO via the receive control and status register. When the control logic determines that a packet is being

transmitted over the Ethernet by another station, the packet is read from the Ethernet into the buffer indicated by the buffer number obtained from the receive buffer available FIFO. After receipt of the packet, the buffer number is moved from the receive buffer available FIFO to the receive complete FIFO. The completed receive buffer number is then read by the processor from the receive complete FIFO via the receive control and status register. The packet is then read by the processor from the packet buffer memory.

All four of the transmit and receive FIFOs are large enough to hold 16 entries. This allows buffers to be used for receive and transmit in any combination.

The CRC circuitry calculates a 32-bit CRC for the transmitted packet and appends it to the end of the packet. During receipt of a packet, the CRC circuitry regenerates the CRC for the packet and compares it to the CRC appended to the received packet. A flag in the receive packet buffer is set if the CRC does not match.

# CHAPTER 4

# PROGRAMMING

The following chapter is pertinent only to those readers who wish to develop their own software drivers. The standard software drivers provided by 3Com for RT-11 and RSX-11M insulate the user from many of these details by presenting a simple data link level interface tailored to the operating system. Documentation for these drivers may be found in Chapter 7.

## 4.1 DEVICE BUFFERS

All packet traffic through the QE Controller passes through the 32K packet buffer memory which appears on the Qbus at an arbitrary 32K byte address boundary selectable by dip switches on the QE control board. To the LSI-11 processor, the buffer memory is indistinguishable from normal RAM memory and can store program code or data if desired.

The memory is divided into an array of 16 buffers of 2K each, all

aligned on 2K byte boundaries. (See Figure 4-1 below.) Buffers are indexed 0-15 with buffer 0 at the low end of QE memory and buffer 15 at the high end. Storing a buffer index into the transmit control register causes the QE Controller to transmit a packet from that buffer. Storing a buffer index into the receive control register causes the QE Controller to receive a packet into that buffer. The results of storing a buffer number into both transmit and receive registers is indeterminate and should not be done.

If a packet buffer index is not stored into either the transmit or receive registers, the buffer will remain untouched by the QE Controller. A buffer employed as a transmit or receive buffer at given time retains

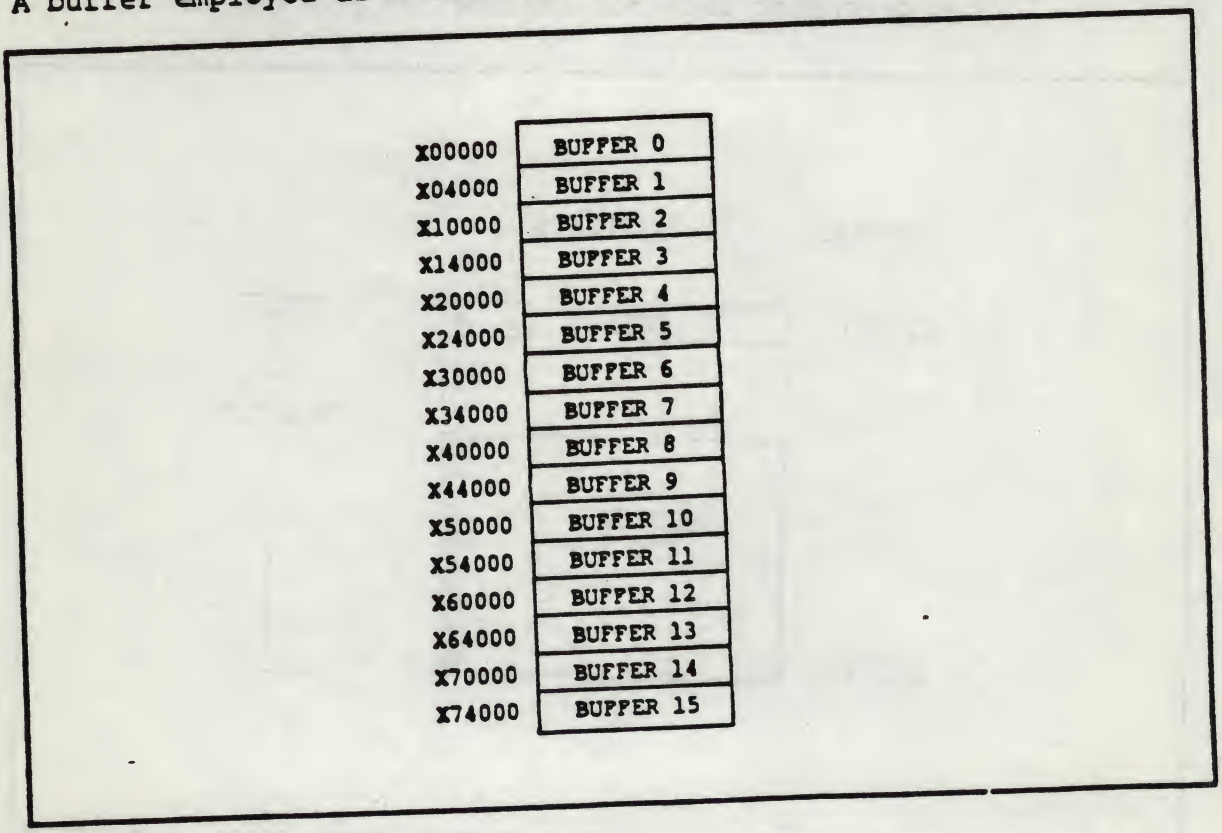



FIGURE 4-1. PACKET BUFFER MEMORY

its role only until the transfer is complete; the buffers used for transmitting or receiving packets can be changed dynamically.

To transmit a packet, the LSI-11 processor must store the packet into an available buffer such that the last byte of the packet coincides with the last byte of the buffer. (See **Figure 4-2** below). The processor must then store into the first 16-bit word of the buffer the byte offset from the beginning of the buffer to the first byte of the packet. (The controller ignores the high order five bits of this word.)

**NOTE:** The packet stored into the packet buffer by the processor does not include the preamble or

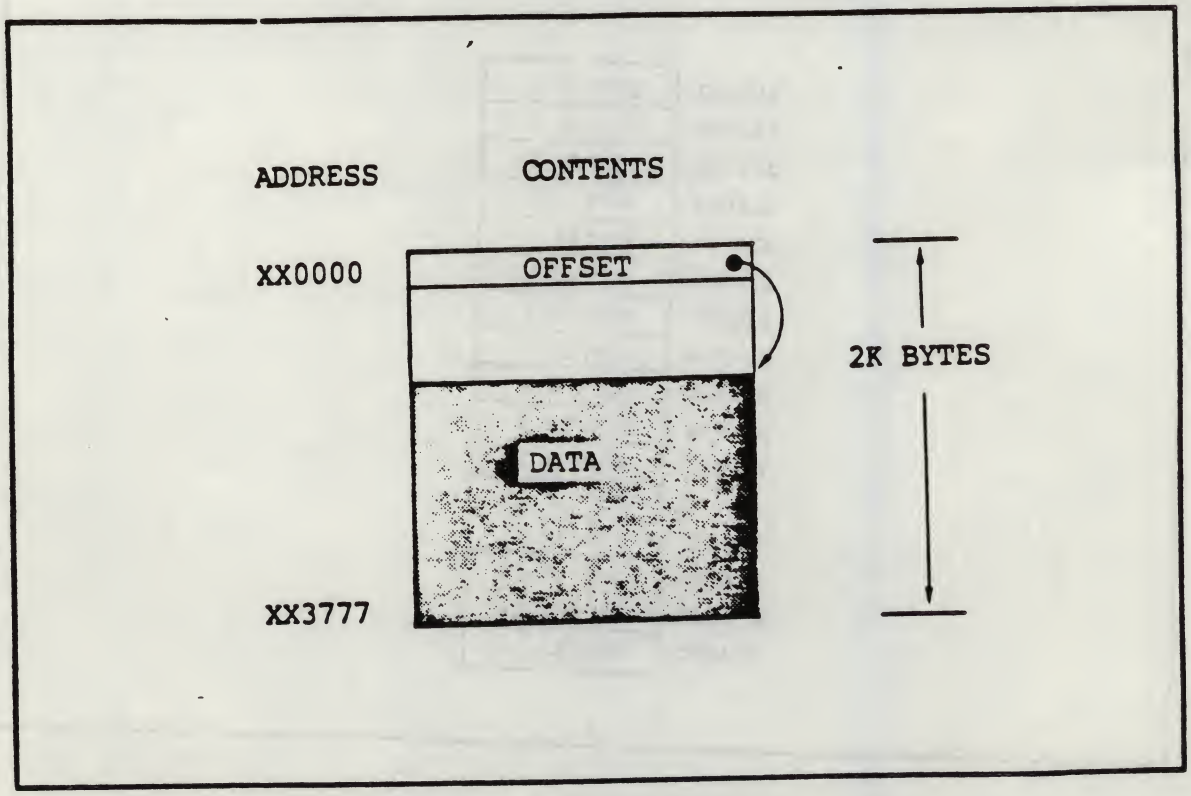



**FIGURE 4-2. TRANSMIT PACKET BUFFER FORMAT**

FCS, but does include the Ethernet data link layer header fields (destination address, source address, and type field) along with the data. Since the maximum legal packet size on the Ethernet is 1514 (excluding the frame check sequence (FCS) field), at least 532 bytes of each transmit packet buffer will always be unused. Conversely, since the minimum legal packet size is 60 bytes (again excluding the FCS field), at most 1986 bytes of each transmit packet will be left unused. It is the responsibility of the driver software to insure that minimum and maximum packet size requirements are observed.

Received packets are stored by the QE Controller into an available receive packet buffer at a fixed offset 528 bytes from the beginning of

the buffer. (See Figure 4-3 below). This offset leaves exactly enough room for a maximum sized packet plus the four FCS bytes plus two guard bytes. After packet receipt, the QE Controller stores into the first word of the packet buffer the byte offset from the beginning of the buffer to the first byte beyond the end of the received data and FCS.

If the stored offset is zero or points to the odd guard byte, the received packet has exceeded the legal maximum size. The high order bit of the offset, if set, indicates an FCS error. In either case, the FCS field that accompanied the received packet can be found immediately trailing the data. The QE Controller does not discriminate against packets smaller than the minimum legal size. The driver software must

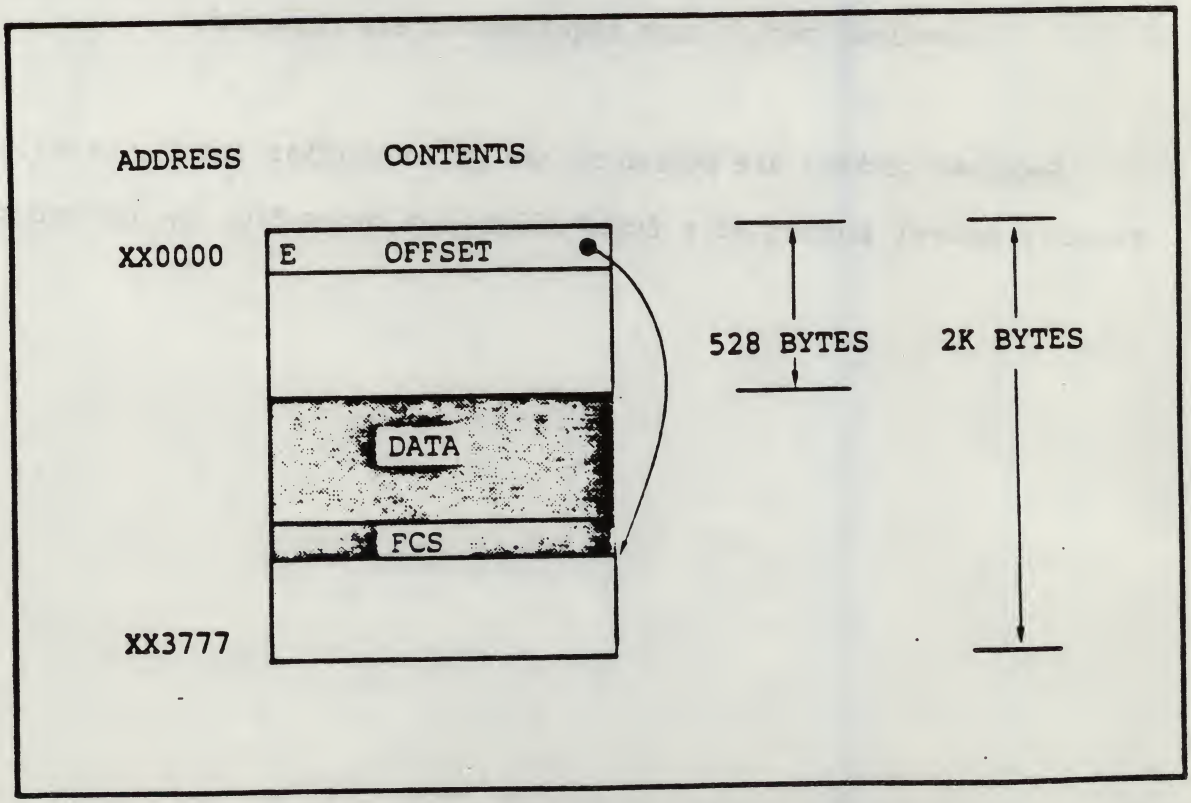



FIGURE 4-3. RECEIVE PACKET BUFFER FORMAT

discard received packets that do not meet the legal size requirements, i.e., offset < 592 or offset > 2046.

## 4.2. DEVICE REGISTERS AND TRAP VECTORS

Each QE uses two words in the I/O page, QERCR and QEXCR, the receiver control register and the transmitter control register respectively, shown below. There are three trap vectors for receive (QERTRP), jam (QEJTRP), and transmit (QEXTRP). The addresses of these registers and trap vectors are set by dip switches on the control board.

| REGISTER | NAME | BUS ADDRESS |
|---|---|---|
| Receive Control Register | QERCR | 111xxxxxxxxxxx00 |
| Transmit Control Register | QEXCR | 111xxxxxxxxxxx10 |

| TRAP VECTOR | NAME | BUS ADDRESS |
|---|---|---|
| Receive Done Trap Vector | QERTRP | 0000000xxxxx0000 |
| Transmit Jam Trap Vector | QEJTRP | 0000000xxxxx0100 |
| Transmit Done Trap Vector | QEXTRP | 0000000xxxxx1000 |

## Receive Control and Status Register

The Receive Control Register (QERCR) controls the receive section of

the QE Controller. (See Figure 4-4 below)

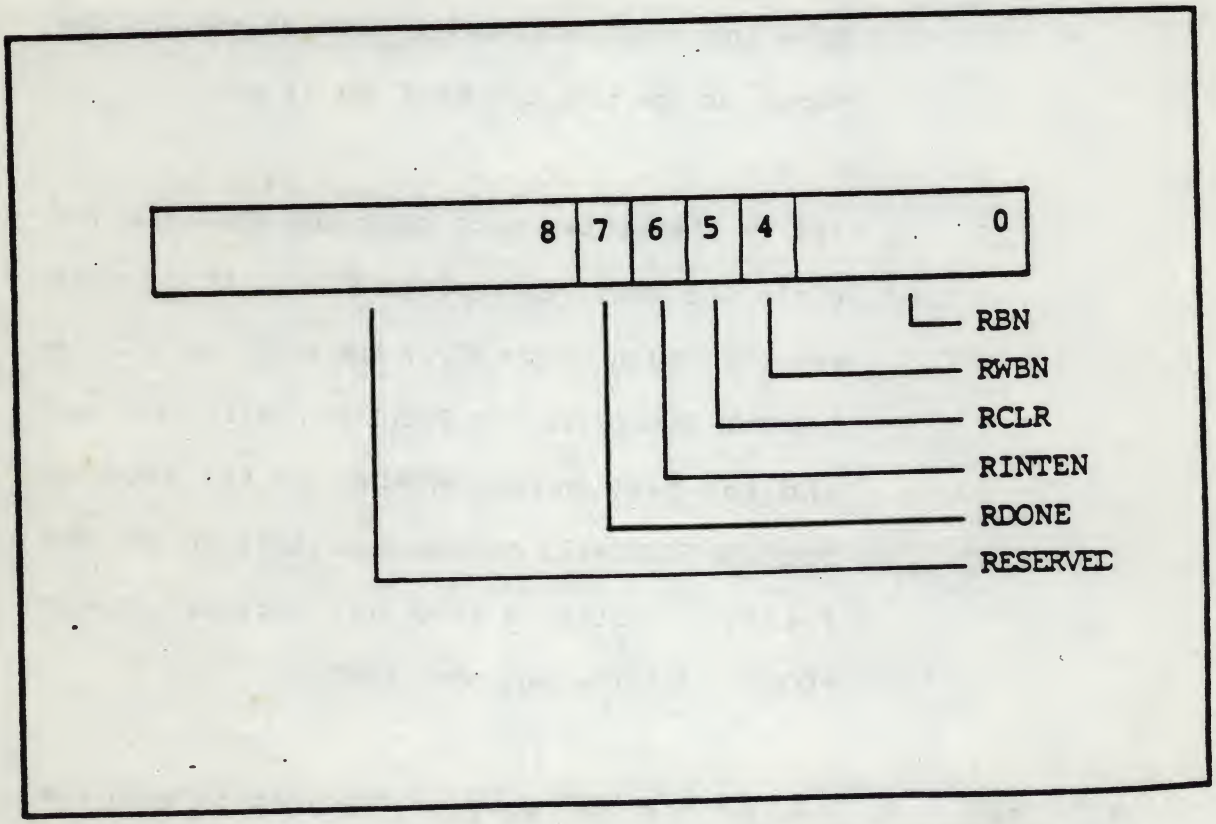



FIGURE 4-4. QERCR RECEIVE CONTROL AND STATUS REGISTER

| BIT | NAME | FUNCTION |
|---|---|---|
| 15-8 | | Reserved for future use. Never write a non-zero value into this field. Returns zero when read. |
| 7 | RDONE | Set by the controller when a receive is done. When set, the RBN field contains the buffer number of the received packet, i.e., the head end of the receive complete FIFO. Writing this bit has no effect. |
| 6 | RINTEN | Set/reset by the software. If set, a receive done interrupt will occur through the QERTRP vector at the time the RDONE bit is set. |
| 5 | RCLR | Set by the software to clear the RDONE flag and cycle the receive complete FIFO. If no more entries remain in the FIFO, the RDONE bit will be cleared, Otherwise the RDONE bit will stay set and the next buffer number in the receive complete FIFO will become available in the RBN field. Writing a zero into this bit has no effect. Returns zero when read. |
| 4 | RWBN | Set by the software in combination with the contents of the RBN field to write a buffer |

number onto the tail of the receive buffer available FIFO. Once a buffer has been submitted to the receiver, it may not be submitted either to the receiver or transmitter until it has been returned by the receiver, i.e., appeared in the RBN field with RDONE set. Writing zero into this bit has no effect. Returns zero when read.

3-0 RBN — Dual purpose receive buffer number field. A write with RWBN set causes the RBN field to be stored into the tail end of the receive buffer available FIFO. A write with RWBN clear has no effect. A read with RDONE set yields the head of the receive complete FIFO in the RBN field. A read with RDONE clear is undefined.

## Transmit Control and Status Register

The Transmit Control and Status Register (QEXCR) is used to control the transmit section of the QE Controller. (See Figure 4-5 below.)

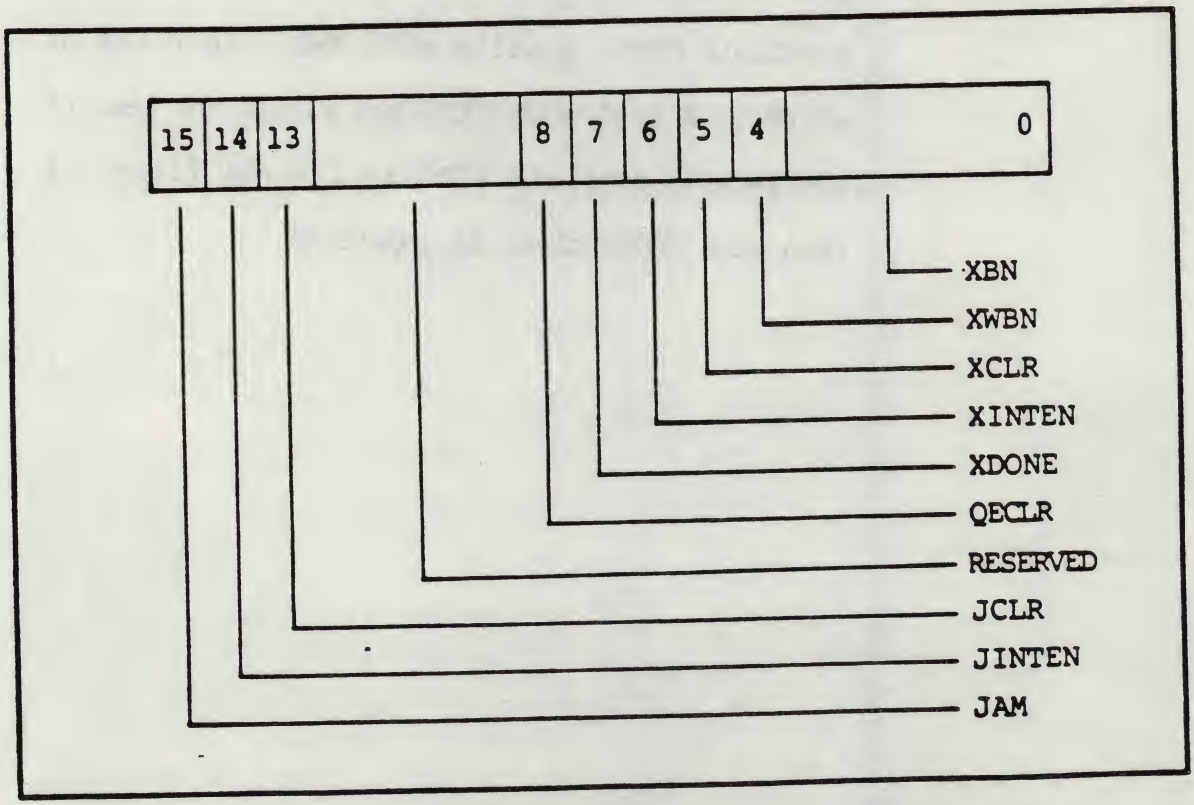



FIGURE 4-5. TRANSMIT CONTROL AND STATUS REGISTER

| BIT | NAME | FUNCTION |
|---|---|---|
| 15 | JAM | Set by the controller whenever a collision is detected during transmission. At the same time the controller transmits a 32-bit jam signal. The JAM flag stays set until the software sets the JCLR bit. While JAM is set, the transmitter is disabled (but the receiver is not). Writing this bit has no effect. |
| 14 | JINTEN | Set/reset by the software. If set, this flag causes a jam interrupt through the QEJTRP vector at the time the JAM bit is set. |
| 13 | JCLR | Set by the software to clear the JAM flag. Enables retransmission of the jammed packet. The processor should delay according to the Ethernet binary exponential backoff algorithm before clearing the JAM bit. Writing a zero into this bit has no effect. Returns zero when read. |
| 12-9 | | Reserved for future use. Never write a non-zero value into this field. Returns zero when read. |
| 8 | QECLR | Set by the software to abort any operations in progress and reset the QE Controller to its |

initial state. A Qbus Init has the same effect.

| | | |
|---|---|---|
| 7 | XDONE | Set by the controller when a transmit is done. When set, the XBN field contains the buffer number for the transmitted packet, i.e., the head end of the transmit complete FIFO. Writing this bit has no effect. |
| 6 | XINTEN | Set/reset by the software. If set, a transmit done interrupt will occur through the QEXTRP vector at the time the XDONE bit is set. |
| 5 | XCLR | Set by the software to clear the XDONE flag and cycle the transmit complete FIFO. If no more entries remain in the FIFO, the XDONE bit will be cleared, otherwise the XDONE bit will stay set and the next buffer number in the transmit complete FIFO will become available in the XBN field. Writing a zero into this bit has no effect. Returns zero when read. |
| 4 | XWBN | Set by the software in combination with the contents of the XBN field to add a buffer number to the tail of the transmit buffer pending FIFO. Once a buffer has been submitted to the transmitter it may not be submitted either to the |

receiver or transmitter until it has been returned by the transmitter, i.e., appeared in the XBN field with XDONE set. Writing a zero into this bit has no effect. Returns zero when read.

3-0 XBN — Dual purpose transmit buffer number field. A write with XWBN set causes the XBN field to be stored into the tail end of the transmit buffer pending FIFO. A write with XWBN clear has no effect. A read with XDONE set yields the head end of the transmit complete FIFO in the XBN field. A read with XDONE clear is undefined.

## 4.3 OPERATION

The transmitter and receiver operate independently, giving the programmer the illusion that the Ethernet is a full-duplex device. In fact, only one packet may exist on the coaxial cable at a time.

The software may supply one or more packet buffers to the transmitter and receiver sections of the QE Controller in any order. Whenever the Ethernet channel is not in use, any and all packets supplied to the transmitter (waiting in the transmit pending FIFO) are transmitted as quickly as possible, separated by the minimum packet spacing of 10.2 microseconds. Packets are transmitted in the order they were supplied to the transmitter and the transmit done indications are returned by the transmitter in the same order transmitted.

Whenever a packet appears on the Ethernet, it is read into a buffer previously supplied to the receiver (waiting in the receive buffer available FIFO). If multiple buffers are waiting in the receive buffer available FIFO then back-to-back packets can be read into successive receive buffers without intervention by the processor. Whenever the receiver buffer available FIFO is empty, packets going by on the Ethernet are ignored. Packet buffers store received packets in the order supplied to the receiver and the receive done indications are returned by the receiver in the same order received.

One aspect of the transmit process, the binary exponential backoff

algorithm, is not implemented by the QE hardware and requires software support. When the controller detects a collision it sets the JAM bit in the transmit control register (QEXCR). This causes a 32-bit jam signal to be transmitted and, if the JINTEN bit is set in QEXCR, an interrupt occurs through the QEJTRP vector. The software is responsible for delaying (backing off) the appropriate amount of time after a collision. After the delay, the software must clear the JAM bit by setting the JCLR bit in QEXCR. This re-enables the transmitter, allowing the same packet to be retransmitted. If a collision happens again, the cycle repeats: the JAM bit comes on, the software detects this event, delays, then restarts the transmitter by setting JCLR.

The delay time is an integral multiple of a slot time (512 bit-times or 51.2 microseconds). The integral multiple is chosen as a uniformly distributed random integer greater than or equal to zero and less than 2k, where k is either the number of retransmission attempts for the packet being transmitted or 10, whichever is less. This algorithm doubles the mean of the delay time each time a collision occurs, ensuring the stability of the Ethernet even under extreme loading.

If the number of retransmission attempts exceeds 15 (probably a transmission subsystem malfunction), an error should be reported.

**NOTE:** If multiple transmit buffers are pending, the procedure for detecting whether successive jam attempts apply to the same or different packets may not be immediately obvious.

# CHAPTER 5

# INSTALLATION AND CONFIGURATION CONSIDERATIONS

## 5.1 INSTALLATION CHECKLIST

___ Before opening the shipping carton, inspect it for damage or water stains, if so,

- Write a brief description of the damage on the bill of lading, and

- Request that the carrier's agent be present when the carton is opened.

- Save the carton and packing materials to show the carrier in case the controller was damaged. The carrier is liable for shipping damage.

___ Unpack the three QE Controller modules gently by removing the foam packing material. Verify that all components are present (See Figure 5-1 on the next page.)

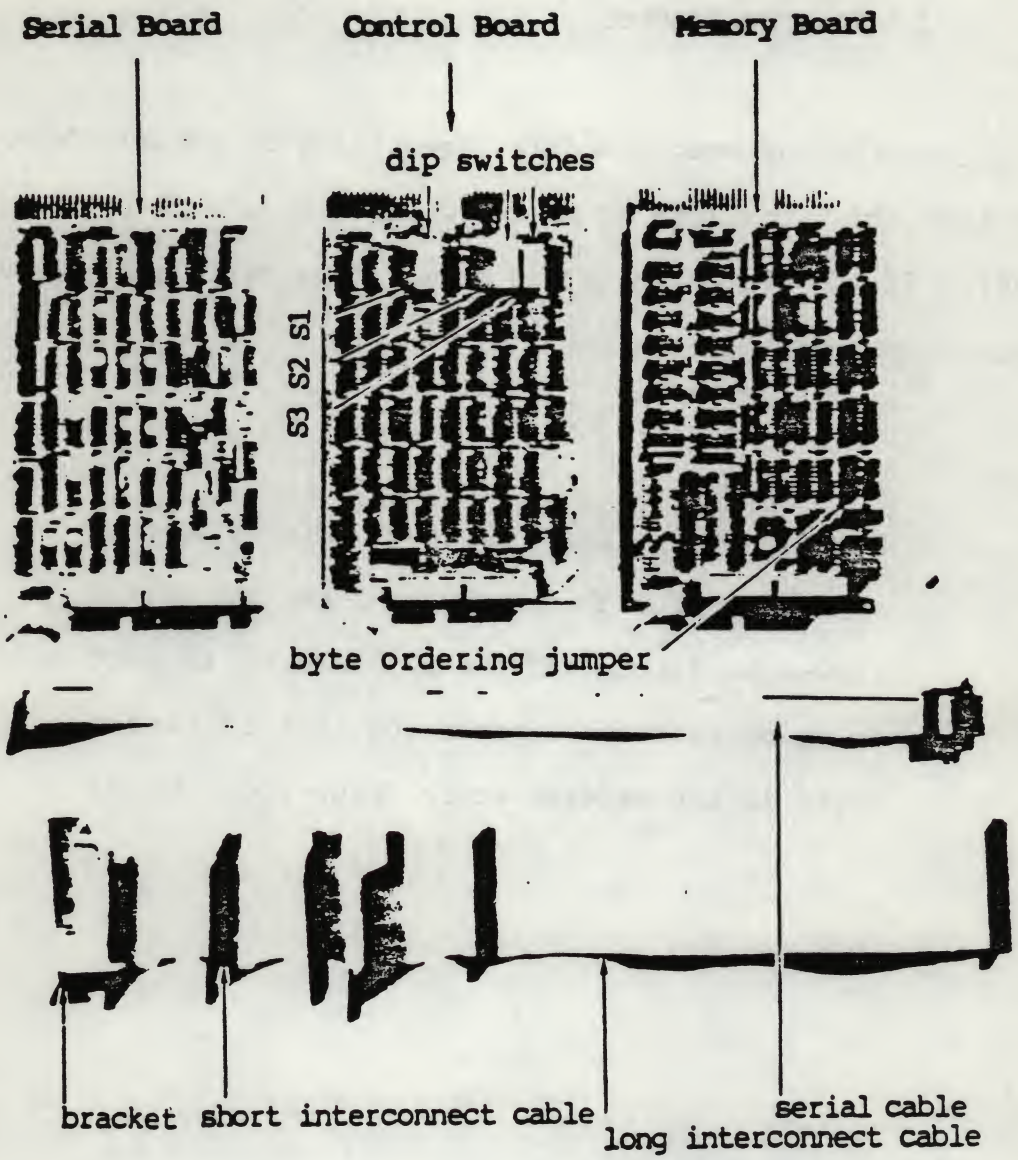



FIGURE 5-1. QE CONTROLLER PARTS

**3 - PC Boards**

**2 - Cables**

1-50 conductor cable with 3 50-pin connectors

1-14 conductor cable with 2 14-pin connectors

**1 - Mounting Bracket**

___ Locate the **memory module**, identified by the word "memory" in white along the bottom edge of the board, to configure the byte ordering jumper. If the QE Controller is going to be operated by an LSI-11 processor, go on to the next step.

The jumper is configured at the factory to address bytes in the proper order for the LSI-11 (low-order first). If the QE Controller is going to be operated by a processor that addresses bytes in the reverse order (high-order first)

remove the jumper (See Figure 5-2 below).

___ Locate the control module, identified by the word "control" in white along the bottom edge of the board, to configure the address option switches. If the addresses configured at the factory are satisfactory, go on the the next step. The factory defaults are:

| Description | Factory default | Switch names |
|---|---|---|
| Control Register Base Address | 764330 | R12 - R02 |
| Trap Vector Base Address | 400 | T08 - T04 |

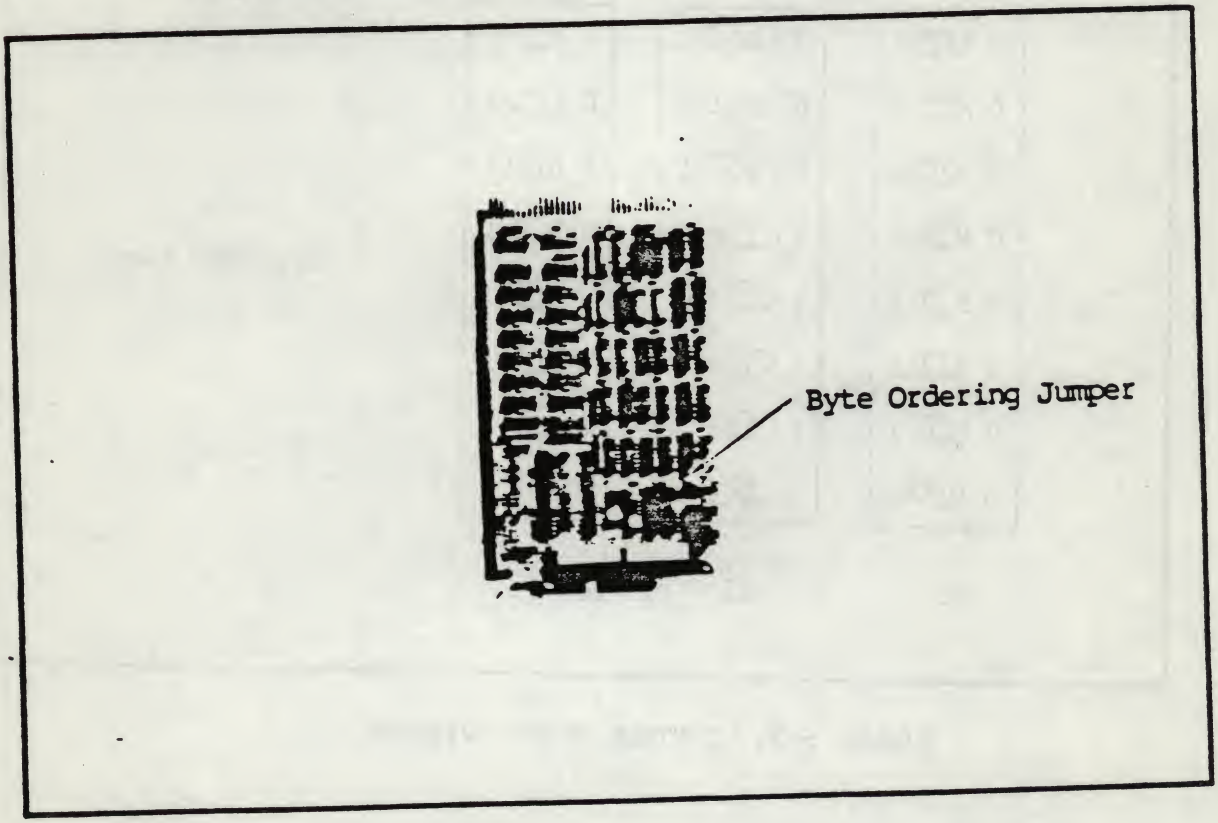



FIGURE 5-2. MEMORY BOARD JUMPER

| | | |
|---|---|---|
| Buffer Memory Base Address | 100000 | M17 - M15 |
| Extended Base Address | 17000000 | M21 - M18 |
| Extended Addressing Enabled | (no) | EAE |

If non standard addresses are desired for any of the above, modify the option switch positions according to Figure 5-3 below.

**Note: The trap vector switches T04 through T08 are in the reverse orientation from the rest of the option switches.**

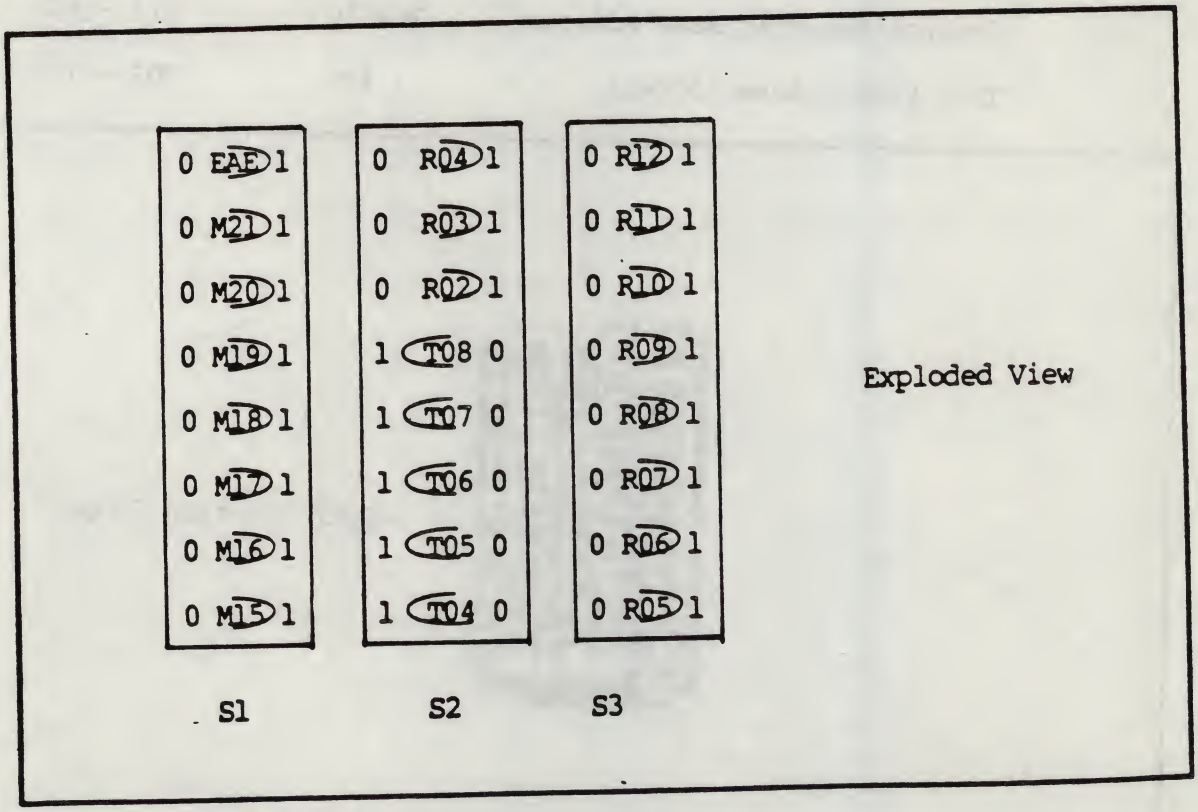



FIGURE 5-3. CONTROL BOARD SWITCHES

The factory default switch settings are shown in Figure 5-4 below.

See section 5.2 for a description of the considerations involved in choosing Qbus addresses for the QE Controller.

___ Turn the DC power to the backplane OFF.

___ Plug the three dual boards into electrically adjacent slots in the Qbus backplane. If the standard short interconnect cable was ordered with the QE Controller, the modules must be grouped together as shown by

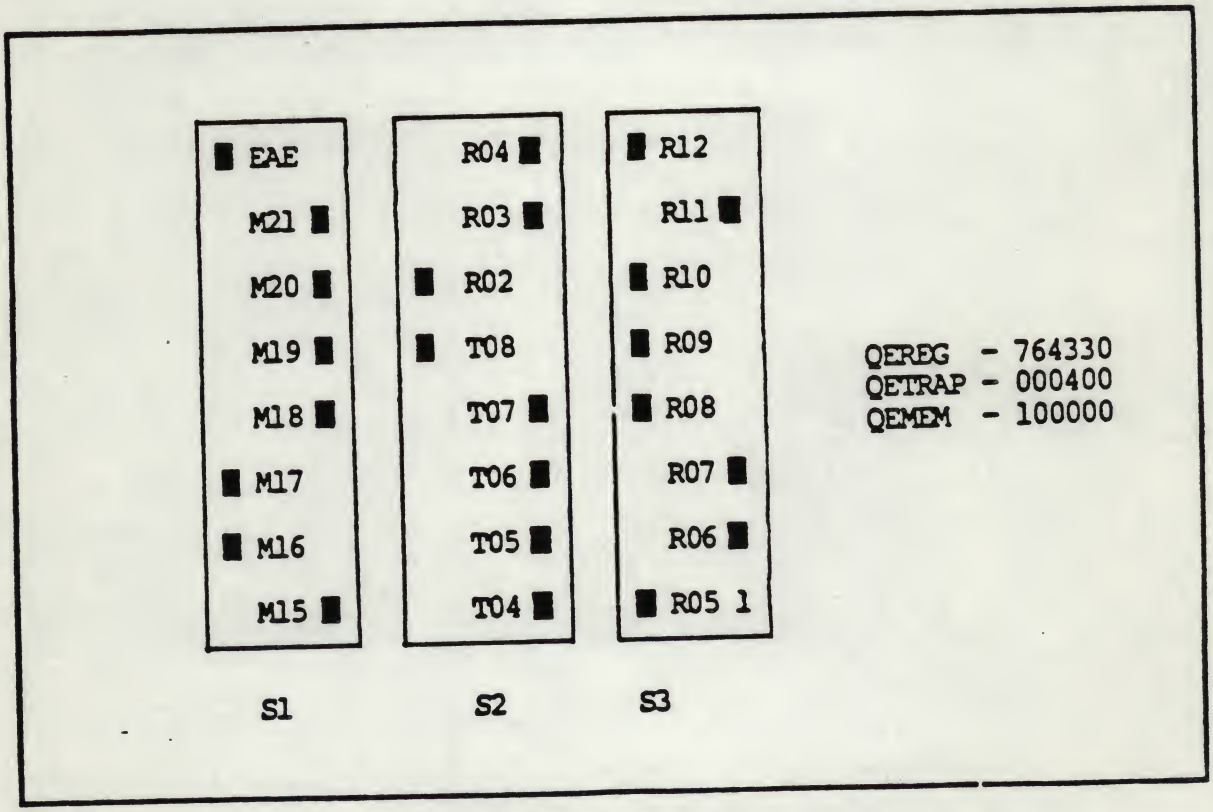



FIGURE 5-4. FACTORY DEFAULT SWITCH SETTINGS

Figure 5-5 on the next page. If the longer (L option) interconnect cable was ordered the modules may be placed in an L configuration as shown by Figure 5-6 on the second page following.

**IMPORTANT. Install the Control Card first, electrically, on the bus because it has the bus grants on it.** The other cards may be installed in any order.

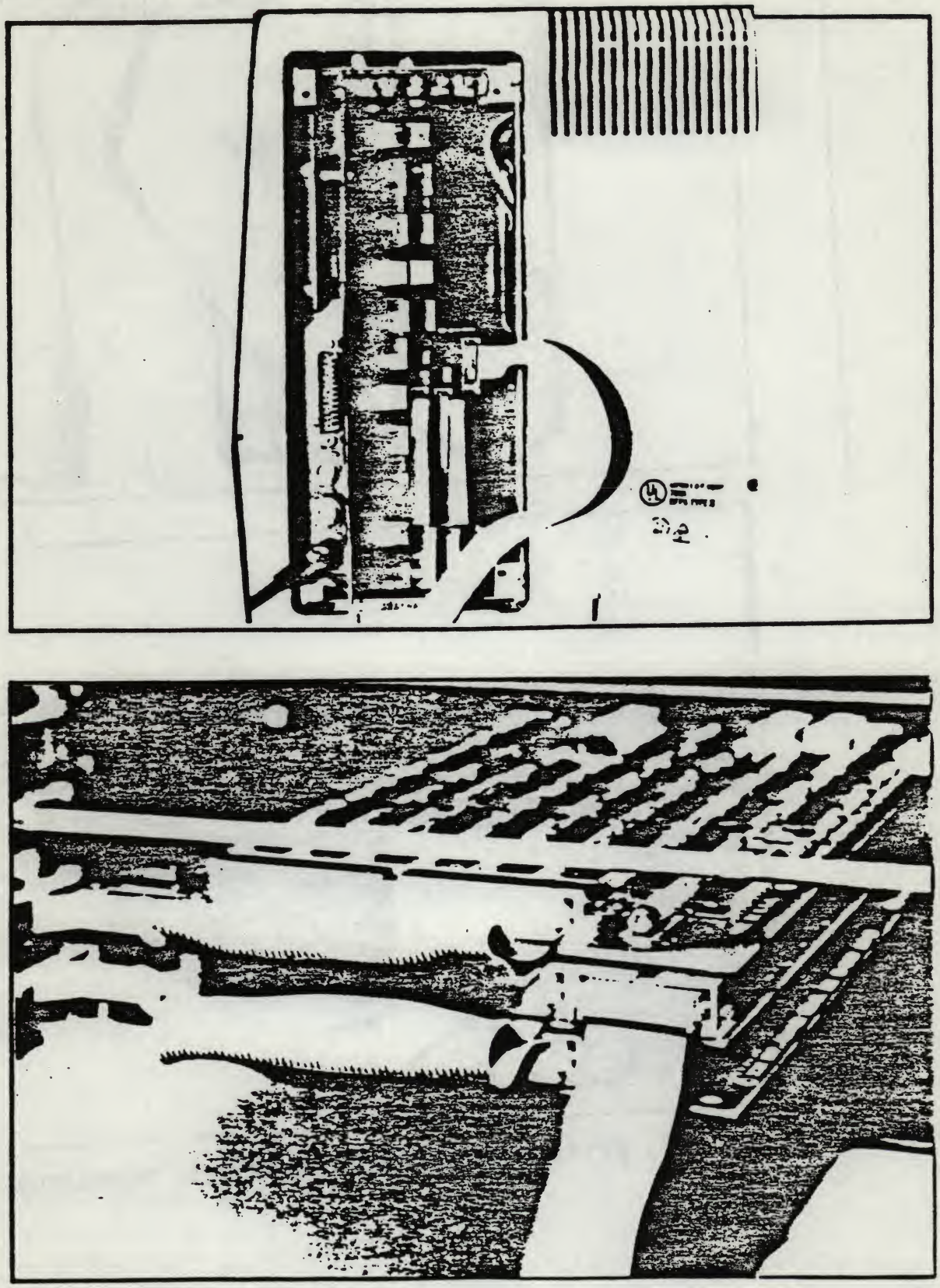


FIGURE 5-5. SHORT INTERCONNECT CABLE INSTALLATION

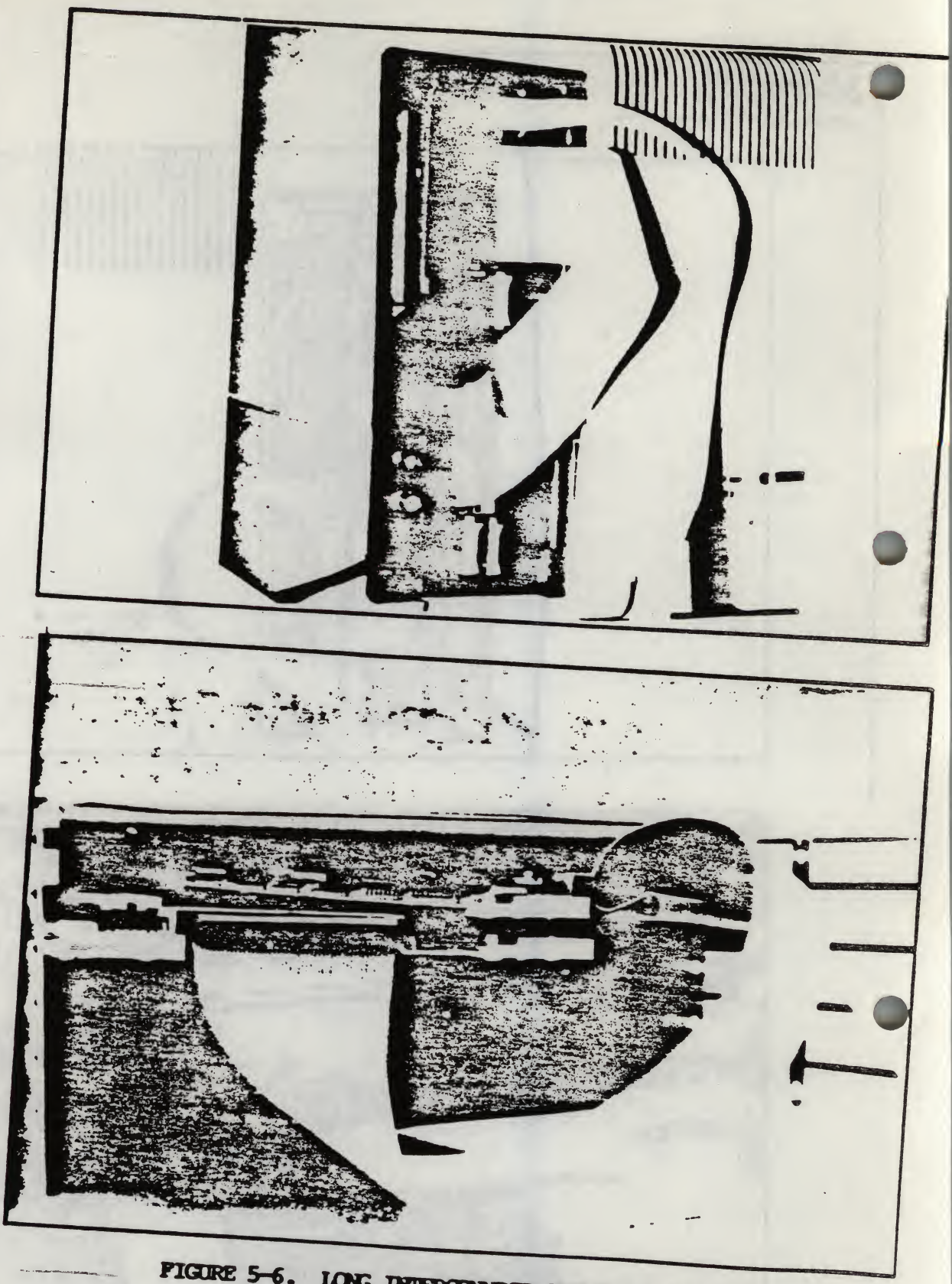


FIGURE 5-6. LONG INTERCONNECT CABLE INSTALLATION

___ Plug the 50-conductor interconnect cable into the 50-pin connector on the back edge of the three modules. Make sure the boards and the interconnect cable are firmly seated.

___ Plug the 14-conductor serial cable supplied into the smaller 14-pin connector on the back edge of the serial module.

___ Install the mounting bracket on the metal chassis somewhere that can be conveniently reached by the serial cable.

___ Attach the other end of the serial cable to the chassis mount and fasten the metal screw-eye protruding from the serial cable connector to the metal chassis mount. Proper grounding of the connector shell in this manner reduces the electromagnetic interference (EMI) caused by the QE Controller.

___ Plug the Ethernet transceiver cable (not supplied) into the transceiver cable connector mounted on the enclosure chassis and you are ready to go.

See Chapter 6 for instructions on running the diagnostic software.

## 5.2 QBUS ADDRESSING CONSIDERATIONS

The first configuration option is the byte ordering jumper on the memory board. Remove the byte-ordering jumper to reverse the order that bytes are addressed when the buffer memory is referenced by the LSI-11 processor: high-order byte first rather than (default) low-order byte first. This allows other Qbus-compatible processors that address bytes in the reverse order to use the QE Controller without swapping the bytes in each word.

The control board has three 8-bit dip switches that specify user-configurable addresses. Eleven switches specify address bits 12 through 2 of the transmit and receive control and status registers. Five more specify address bits 8 through 4 of the three trap vectors. Three switches specify address bits 17 through 15 of the 32K buffer memory. For LSI-11 processors that support 22-bit extended addressing on the Qbus, an additional 4 switches are provided to specify address bits 21 through 18 of the buffer memory. The last switch enables extended addressing; when turned off the four extended address dip switches are ignored.

The 32K buffer memory can be assigned on any 32K memory boundary using the 8 dip switches just mentioned. The LSI-11/2 and the LSI-11/23, without a memory management unit (MMU), have only 16 bits (64K bytes) of physical address space. In such situations the 32K buffer memory may be configured into either the lower or upper half of the 64K address space. If the upper configuration is chosen, the last 8K bytes of the QE memory will be obscured by the I/O page and will therefore be inaccessible. The

remaining twelve 2K buffers will function properly, however, more than enough for typical applications.

Given an LSI-11 with an MMU, the buffer memory can be assigned to any 32K bank of memory within the 18 bits (256K bytes) of physical address space. If the MMU 22-bit addressing mode is enabled and a 22-bit LSI-11 backplane is used the QE buffer memory can even be located outside of the lower 256K.

## 5.3 ETHERNET ADDRESS CONSIDERATIONS

Each station on an Ethernet has a 48-bit Ethernet address associated with it that is unique across all Ethernets in the universe. The station address is assigned by the manufacturer of the station. For stations assembled from components from multiple manufacturers, the assignment of Ethernet addresses is ambiguous. This section describes the conventions to be followed to maintain unique Ethernet addresses under various configurations of components.

Each QE Controller manufactured by 3Com is shipped with a unique Ethernet address. The address is printed on the serial module in indelible ink. For stations that contain only a single QE Controller, the station address is the one supplied with the QE Controller. For stations that contain multiple QE Controllers, such as gateways, one of the controllers should be arbitrarily selected to contribute its Ethernet address as the station address. The software address recognition in the

station must be programmed to respond to the chosen Ethernet address in all the QE Controllers attached.

If it becomes necessary, due to faulty hardware, to swap boards between stations at a user site, it is the user's option either to swap the Ethernet address in software to follow the board, or to dissassociate the board from the address in software. The advantage of the former is that the address recognized by the software always matches the address printed on at least one of the controllers in the station. The advantage of the latter is the ability to swap hardware around without changing any software.

If a board is sent back to the factory for repair, the board that comes back may not have the same Ethernet address printed on it. The customer has the option of using either the new Ethernet address or the old. Both addresses are allocated to the customer and are under the customer's control.

It is the customer's responsibility to manage the allocated Ethernet addresses which can be assigned in any manner as long as no Ethernet addresses are duplicated on more than one station. The recommended practice is to add each QE Controller Ethernet address to a site-wide pool of addresses maintained independently of the hardware. That way, QE hardware can be moved around among stations or sent back to 3Com for repair without affecting any software.

## 5.4 Q-BUS PERFORMANCE CONSIDERATIONS

The QE controller 32K byte buffer memory, being dual-ported, is slower than other commercially available Qbus memories. This difference in basic memory cycle time maybe significant in system configurations depending upon faster memories. In particular, high-speed disk drives using unbuffered disk controllers may require a DMA transfer rate which exceeds the capabilities of the QE memory. A DMA-TYPE CONTROLLER REQUIRING A BASIC MEMORY CYCLE TIME OF LESS THAN 800 NANOSECONDS SHOULD NOT READ OR WRITE THE QE CONTROLLER MEMORY. The following table details the QE controller memory timing.

DATI TYPICAL CYCLE TIME (READ)

| | LSI-11/23 | LSI-11 |
|---|---|---|
| BSYNC to BDIN | 130 ns | 190 |
| BDIN to BRPLY | 350 | 350 |
| BRPLY to BDIN inactive | 220 | 220 |
| BRPLY inactive to BSYNC synchronization time for dual-port | 300 | 300 |
| | 800 | 800 |
| BUS Cycle Time (BSYNC to BSYNC) | 1800 ns | 1860 ns |

DATO TYPICAL CYCLE TIME (WRITE)

| | LSI-11/23 | LSI-11 |
|---|---|---|
| BSYNC to BDOUT | 200 ns | 285 |
| BDOUT to BRPLY | 350 | 350 |
| BRPLY to BDOUT inactive | 285 | 285 |
| BDOUT inactive to BSYNC | 300 | 300 |
| synchronization time for dual-port | 800 | 800 |
| BUS Cycle Time (BSYNC to BSYNC) | 1995 ns | 2020 ns |

# CHAPTER 6

# TROUBLESHOOTING, SERVICE AND REPAIR

## 6.1 OVERVIEW

ECTEST software is a set of 12 functional tests for the QE Ethernet Controller. These tests run without the assistance of an operating system, in an LSI-11 configured with a console terminal and one or more QE Controllers.

The memory and control boards are directly testable with software. The serial board, not programatically accessable, is tested in concert with the other two boards while exchanging packets with a remote host.

ECTEST is loadable directly from secondary storage using a ROM monitor; it includes the software that implements an Ethernet echo-server.

These tests should be run when the Controller is first installed and whenever verification of its operation is necessary. A successful completion of these tests indicates the proper operation of the following QE Controller components and functions:

- The address path, data path, and data retention

characteristics of each memory cell on the 32K byte memory board

- The transmit and receive control registers together with the four internal FIFO registers on the control board

- The interrupt hardware associated with transmitting and receiving packets and detecting collisions

- The parallel-to-serial and serial-to-parallel conversion and CRC calculation hardware on the serial board

- The overall ability to transmit and receive back-to-back Ethernet packets over the network

The individual tests are combined into a single program that is loaded into a LSI-11 with the ROM monitor and boot program. The tests are run in succession with results displayed on the console device. A minimum of user intervention is required. ECTEST requests the hardware locations of the buffer memory, control registers, and trap vectors; they may be defaulted. The test may be paused and resumed upon encountering an error. Also, the number of passes through the set of tests and the particular tests in a pass may be changed.

Verification that entire packets can be transmitted and received may be performed in one of two ways:

1. If a single QE Controller is installed in the LSI-11 backplane, the assistance of another Ethernet station acting as an echo-server is required. Each packet that is received by this echo-server is immediately returned with the source and destination addresses reversed. The software that implements the echo-server is included with ECTEST.

2. If more than one QE Controller is present in the LSI-11 backplane, each controller may echo packets off the above echo-server or communicate with another controller in the backplane. The second method offers the ability to verify the reception of back-to-back packets.

## 6.2 FUNCTIONAL TESTS

Following are descriptions of the twelve tests performed by ECTEST:

### 6.2.1 Memory Write/Read - Address

This test makes an initial pass through the entire 32K byte memory, writing the location of each cell into itself. It pauses for four milliseconds for retention and then reads each location in the reverse direction comparing the contents of each memory location with its address. A second iteration is made, reversing the direction for writing and reading the memory. The purpose of this test is to verify that the memory address lines are working properly.

### 6.2.2 Memory Write/Read - Floating 1's and 0's

This test makes sixteen passes through the entire 32K byte memory, writing and then reading a pattern that propogates a 1 through the sixteen bit positions of each word. Upon completion, the test is repeated, propogating a 0 through each bit position. The purpose is to verify that the memory data lines are working properly.

### 6.2.3 Memory Write/Read - Random Data

This test makes a pass through the 32K byte memory writing random data patterns into each memory cell. A second pass is made, reading each

cell and comparing it to data produced by the same random seed. A total of 16 iterations of this two pass sequence is made. The purpose of this test is to search for pattern sensitive conditions and to simulate the operation of the RAM under normal conditions.

### 6.2.4 Read-Modify-Write / Byte Operations

This test checks read-modify-write instructions, i.e., swap and increment, on a memory board location. Also, the CPU is tested to access individual bytes on the memory board using byte instructions. The least significant byte in a memory cell is written and then read. The adjacent byte is examined for any disturbance. The same test is repeated for the most significant byte.

### 6.2.5 Transmit a Single Packet

This test creates a single maximum-sized packet and transmits it. The control register is written with the packet's buffer number (XBN) and the write buffer number bit (XWBN) is set. The test then loops, checking both the jam bit (JAM) for a collision and the done bit (XDONE) for successful transmission. If the XDONE bit responds, the clear bit (XCLR) is set and the transmitted buffer number is read from XBN and verified. The XDONE bit is again checked to assure that it has been cleared.

If JAM responds, the clear bit (JCLR) is set and the test continues to loop, checking JAM and XDONE. In either case, a timeout occurs after

two milliseconds indicating an error in transmission. Two milliseconds is approximately twice the time required to send a maximum-sized packet.

This test checks the control logic used for transmitting packets, returning status, and manipulating the transmit FIFOs.

### 6.2.6 Transmit Multiple Packets

This test transmits a packet in each available memory buffer. The size of the packet is varied for each buffer. The same collision and timeout constraints apply as in the above test. The transmission of variable-sized packets and the use of each buffer number are tested.

### 6.2.7 Transmit Back-to-Back Packets

This test prepares all buffers and transmits them back-to-back. The same collision and timeout constraints apply as stated above. This test loops, checking the XDONE and JAM bits. As the XDONE bit responds, each transmitted buffer number is read from XBN by clearing the register with XCLR. The check is made for each buffer number appearing in the transmission completion FIFO in the correct order.

### 6.2.8 Transmit Under Interrupt Control

Both the transmit done and jam interrupts are enabled by setting the XINTER and JINTER bits in the transmit control register. A single packet

is transmitted and the test loops waiting two milliseconds for an interrupt. If the jam interrupt responds, the test clears the jam bit (JCLR) and resumes counting at zero. When the transmit done interrupt occurs, the done bit (XDONE) is cleared and the transmit completed buffer number is read from the XBN and verified. This test checks the operation of the interrupt control logic.

### 6.2.9 Identify an Echo-Server Station

The remaining tests require the assistance of a separate transmitting station. By default, this test declares several receive buffers and sends a broadcast packet on the network. If one or more echo-servers are available, they will respond with a return packet. The first server is chosen to complete the tests. If an echo-server does not respond, the test will loop, retransmitting the broadcast packet after two seconds, for a total of five iterations.

Alternatively, the default may be changed to indicate that an echo-server will NOT be present. This change assumes that more than one controller is installed in the common backplane. As each controller is tested, one of the remaining controllers will be used as the separate transmitting station. This configuration permits more rigorous testing including the reception of back-to-back packets. When the default condition is changed as described above, this test is always successful.

### 6.2.10 Receive A Single Packet

This test declares several receive buffers and waits to receive a maximum-sized packet. If an echo-server is present, a maximum-sized packet is transmitted. Otherwise, an alternate controller in the backplane transmits the packet. It then loops, checking the receive done bit (RDONE) waiting for the packet to return. When the RDONE bit responds, the clear bit (RCLR) is set and the receive buffer number (RBN) is read from the receive control register. If the packet is the expected packet, the data in the buffer is verified to be the same data sent.

Because the address recognition of packets is done in software, all packets on the network are received. Any received packets not destined for this station are discarded. Also, the test will timeout after one second if the packet has not been returned. This test checks the control logic used for receiving packets, returning packets, returning status, and manipulating the receive FIFOs. A round trip of the data is also observed when an echo-server is present.

### 6.2.11 Receive Variable-Sized Packets

This test receives 512 packets varying in size from 142 bytes to 1166 bytes. Each packet is first formatted and then transmitted either by the controller under test if an echo-server is used, or an alternate controller if not. The receipt of variable-sized packets and the use of all buffer numbers by the control logic is verified. A check is made for missing or out-of-sequence packets and an error reported when a packet is

not received within a half a second.

### 6.2.12 Receive Under Interrupt Control

Depending on whether an echo-server is used, this test declares either eight (echo-server) or sixteen receive buffers (no echo-server). Interrupts are enabled and the controller waits to receive a burst of either eight or sixteen packets. If an echo-server is used, the remaining eight buffers are used to transmit the packets. Otherwise, sixteen packets are transmitted from an alternate controller in the same backplane. This process is repeated 256 times.

As each packet is received, its buffer number is checked for proper ordering and its data for consistency. An error is reported if a packet fails to appear after half a second. Using an alternate controller as a transmitter, the ability of the controller under test to receive back-to-back packets is demonstrated.

## 6.3 SYSTEM CONFIGURATION

ECTEST will run in a number of different configurations with the following minimum configuration:

- An LSI-11 processor with the EIS (Extended Instruction Set)
- A console device wired to locations 177560-177566 in memory
- A minimum of 60K bytes of memory (32K are provided by the QE)
- A device from which to load ECTEST

From one to six QE Controllers may be tested successively using a common LSI-11 backplane. The LSI-11 processor board must have the proper memory mapping hardware for this to be possible.

## 6.4 ETHERNET CONTROLLER TEST (ECTEST)

Running ECTEST from the console terminal requires several

interactive steps. First, ECTEST must be loaded from secondary memory into main memory using the ROM monitor.

**WARNING**

**THE FOLLOWING COMMANDS HALT THE PROCESSOR!** Be sure to terminate the system gracefully - because any executing program will disappear.

If an "@" symbol is not already present, press "BREAK"( or the required sequence to communicate with the ROM monitor). Depending on where the ECTEST program is stored (tape or diskette) the proper sequence of commands or register writes must be executed to load the boot program from block zero and started at location zero. After loading and starting the boot program, the following procedure must be followed:

**From Tape:**

```
tm(0,2)              -load the second file
<wait for tape>
0g                   -execute
```

**From Floppy Diskette:**

```
ectest               -load the file
<execution proceeds>
```

Upon execution, ECTEST will ask for the type of terminal connected to the system:

```
Terminal type: "y" or "RETURN"):  ADM?
```

Responding with a "y" initializes the test software to generate the appropriate escape sequences for positioning the cursor on on an ADM-type CRT terminal. If the console terminal is not an ADM-type terminal, type "RETURN" to display the next choice. This process will continue through all choices including "other" corresponding to a terminal with no cursor control until a "Y" is typed.

Following this question, the main screen will display:

```
                3COM ECTEST V1.0
                EC:  01      Pass: 001
Vector: 000400   Memory: 00100000   Control: 164330


Press: "c" to change parameters, "e" for echo-server, RETURN to start
```

Typing "c" lets you change the parameters displayed at the top of the screen. Because ECTEST is capable of testing from one to 16 QE Ethernet Controllers installed in the same backplane, several preliminary

questions are asked:

**Number of controllers (def: 1)?**

A single character from one to six is expected; other characters, including a carriage return, default to testing one controller. The next question is:

**Will an echo-server be present?**

The answer to this question determines whether or not to use an echo-server for tests 10,11, and 12 when receiving. It is only meaningful to reply "N" when the number of controllers to be tested is greater than one. If an echo-server is not present, an alternative controller in the same backplane must be used as a transmitter.

The next question requests the tests to be run for each pass:

**Tests to run (def: 1,2,3,...12)?**

Any combination of the numbers one through twelve may be typed, separated by blanks. A maximum of twelve tests can be specified. Simply responding with a carriage return indicates tests one through twelve in ascending order.

Next, the cursor will be positioned at the first changeable value in

the parameter list. If a carriage return is typed, the entire list will be defaulted. Otherwise, a new value may be entered and subsequent parameters of the QE changed. Initially, each controller is given a set of default values:

| QE | Trap Vector | Memory | Control Registers |
|---|---|---|---|
| #1 | 0400 | 0100000 | 0764330 |
| #2 | 0420 | 0200000 | 0764334 |
| #3 | 0440 | 0300000 | 0764340 |
| #4 | 0460 | 0400000 | 0764344 |
| #5 | 0500 | 0500000 | 0764350 |
| #6 | 0520 | 0600000 | 0764354 |

As the parameters for each controller are either changed or defaulted, the memory and control registers are verified to exist. If either of them do not exist, an error message is printed and a request is made for the proper values. Once all the parameters for each QE have been supplied, testing begins automatically.

If the current station is to be the echo-server "e" should be typed. The following message will appear:

```
3COM ECHO-SERVER
```

Each packet received and recognized is returned to the sender. The

station configured as an echo-server will continue to echo packets sent to it until a "DELETE" is typed. Otherwise, typing a carriage return automatically initiates the set of twelve tests. The following screen will be displayed:

```
                    3COM ECTEST V1.0

                EC: 01      Pass: 0001

Vector: 000400   Memory: 00100000   Control: 164330

        1. Memory Address       SUCCESS
        2. Memory Data          SUCCESS
        3. Memory Random        SUCCESS
        4. W_M_W / Byte         SUCCESS
        5. Tx 1 Packet          SUCCESS
        6. Tx N packets         SUCCESS
        7. Tx B-B packets       SUCCESS
        8. Tx Interrupts        SUCCESS
        9. Echo Identify        SUCCESS
        10. Rx 1 Packet         SUCCESS
        11. Rx N packets        SUCCESS
        12. Rx Interrupts       SUCCESS

(This line displays requests and messages)

Error: (This line displays errors)
```

As each test is run, its description is displayed on the terminal. On completion, either the message "SUCCESS" or "FAILURE" will follow this description. If the test fails, the number of errors will be displayed to the right of the word "FAILURE". If a fatal error occurs (testing unable to proceed), testing will terminate. All errors may be displayed once testing has terminated by typing a carriage return. The interpretation of errors is described in section 6.5.

There are several special characters that may be typed from the terminal. A control-s suspends output to the console. A control-q resumes the output. A "DELETE" character terminates all testing.

## 6.5 ERROR MESSAGES

Two types of error messages are generated by ECTEST.

The first type of error message is in response to an invalid input value typed from the console when initializing ECTEST. Most of these messages are self-explanatory and either default to an assigned value or permit the user to re-type the input. Two of these messages are:

**Error: unable to access controller memory**

**Error: unable to access control registers**

In these cases, either the controller memory or the control registers were inaccessible. The parameters for this QE Controller should be re-examined and changed.

The second type of error message occurs while the QE Controller is being tested. Rather than printing the message to the screen, the error together with other pieces of relevant data, are written into a buffer area in memory. Once testing is complete, either normally or through a fatal error, these errors may be displayed by typing **RETURN**. For CRT-type terminals, typing "RETURN" displays subsequent pages of error messages if they continue beyond a single screen.

The following table describes the possible errors:

| No. | Severity | Description | Data 1 | Data 2 | Data 3 |
|---|---|---|---|---|---|
| 1 | fatal | non-existent memory | mapped mem location | 1st QE mmu addr reg | |
| 2 | non-fatal | incorrect value | mapped mem location | expected value | actual value |
| 3 | non-fatal | frame check error | packet head word | buffer number | |
| 4 | non-fatal | incorrect buffer number | expected buffer no. | actual number | |
| 5 | non-fatal | incorrect memory pattern | mapped memory location | expected pattern | actual pattern |
| 6 | non-fatal | read-mod-write error | mapped memory location | expected value | actual value |
| 7 | non-fatal | low byte error | mapped memory location | expected value | actual value |

| | | | | | |
|---|---|---|---|---|---|
| 8 | non-fatal | high byte error | mapped memory location | expected value | actual value |
| 9 | fatal | no transmit buffer available | | | |
| 10 | non-fatal | transmit timeout | transmit buffer no. | packet data size | |
| 11 | non-fatal | incorrect xmit buffer no. | expected buffer no. | actual buffer no. | packet data size |
| 12 | non-fatal | receiver timeout | transmit buffer no. | packet data size | |
| 13 | non-fatal | maximum jam retries | maximum retries | | |
| 14 | fatal | echo-server unavailable | | | |
| 15 | non-fatal | incorrect receive data | receive buffer no. | expected value | actual value |

## 6.6 TROUBLE SHOOTING

In the event errors occur during the execution of the ECTEST consider the following suggestions:

Symptom:

Memory errors in tests 1, 2, 3, or 4.

Check:

Proper seating of the memory card in the backplane.

Proper installation of interconnect ribbon cable to all connectors, remove - check pins - reinsert.

Correct control board dip-switch setting for memory.

Loose jumper connector controlling byte ordering on memory board.

Symptom:

Tests 5, 6, or 7 return errors.

Check:

Proper connection of the serial cable to both the serial board and the transceiver interface cable.

Correct control board dip-switch settings for the memory and control registers.

Symptom:

Test 8 hangs and the processor halts (the run light goes out).

Check:

Proper ordering of cards in the backplane to pass bus grants from board to board.

Correct control board dip-switch setting for the trap vectors.

Symptom:

Test 9 returns a fatal error - unable to find an echo-server.

Check:

Proper connection of the serial cable to both the serial board and the transceiver interface cable.

Availability of an echo-server if one is to be used.

Correct control board dip-switch setting for memory.

Symptom:

Tests 10, 11, or 12 return errors.

Check:

Proper seating of all controller boards in the backplane.

Correct control board dip-switch settings for trap vectors, memory, and control registers.

Proper connection of serial cable to both the serial board and the transceiver interface cable.

## 6.7 WARRANTY AND REPAIR POLICY

Any 3Com product which fails within 90 days from date of shipment will be repaired or replaced by 3Com free of charge providing, in the opinion of 3Com, the product has not been subject to improper electrical, mechanical, or thermal stress. The customer should send the defective item to 3Com stating the nature of the fault and quoting the date and invoice number of the original shipment. Freight charges from customer to 3Com must be paid by the customer. Freight charges from 3Com to U.S. customers are paid by 3Com using UPS. Foreign customers must pay freight both ways. Replacement or repair under warranty will normally be completed within seven days of receipt at 3Com.

After 90 days, a factory repair service is available for the 3C200 Q-bus Ethernet Controller. The charge is fixed at $250 for each controller regardless of the extent of the repairs concerned. If testing of the controller shows no failure, or if the controller has been modified or damaged by the customer attempting repairs or if it has been subject to improper electrical, mechanical, or thermal stress, it will be returned to the customer in the same condition as received and a $100 charge assessed. Controllers repaired under the fixed-price repair service will be warranted for 90 days from the date of shipment of the repaired controller.

In order to achieve a quick turnaround, 3Com may ship a controller different from that received. In all cases, the controller shipped from

3Com will be given a new Ethernet address to avoid any possible duplication of addresses. The customer may use the Ethernet address originally shipped with the controller, or with the repaired controller, or both.

# CHAPTER 7

## RT-11 DRIVER

The device driver described here works with version four of RT-11. A working knowledge of the RT-11 Software Support Manual and the RT-11 Programmer's Reference Manual is assumed.

The RT-11 Software Support Manual details the installation and structure of device drivers.

The monitor calls and the theory of I/O programming are described in the RT-11 Programmer's Reference Manual.

To establish a correspondence between an RT-11 channel and the QE, use the .LOOKUP monitor call using the identifier "QE" to specify the QE. The following example opens the QE on channel zero.

```
        .LOOKUP #LKBLK, #0, #QENAME
                BCC QEOK
        TYPE <QE not available>          ;ERROR!
                HALT
                                         ;OK
QEOK:   . . . . .
                . . . . .
QENAME: .RAD50 /QE/
LKBLK:  .BLKW 3
```

The driver, as supplied, transmits packets one at a time. When called to transmit a second packet, the driver hangs until the previous transmit completes. When collisions occur, the driver retransmits the same packet a maximum of sixteen times, alerting the client with an error. Errors also result for packets too large and for illegal .SPFUNs.

All packets going by on the ether are ignored until the client try to read from the QE. As many as three read requests may be pending at any one time to catch back-to-back packets. If no read requests are pending, all packets going by on the ether are lost. The driver performs address recognition, frame check sequence (fcs) checking, and runt filtering.

The driver accepts broadcast packets as well as the ones addressed specifically to the network node.

When the driver accepts a packet, it writes the byte count (guaranteed to be non-zero) in the first word of the packet buffer followed by the packet. To detect a packet in this buffer, write a zero in the first word of the receive buffer before doing a read, then wait for a non-zero value. If the packet buffer is not big enough for the packet, the packet is lost and an error results.

With the exception of address recognition on receive, the driver does not alter or examine any data in the packets. In particular, the

client must provide the Ethernet header: source, destination, and type in the buffer on transmit. On receive, the client must examine the type field to determine the format of the rest of the packet.

The driver implements .SPFUN code 370 octal to return the network address of the controller. The call should be as follows:

.SPFUN area, chan, func, buf, wcnt, blk, crtn

buf and wcnt specify a 31 word area where the driver returns the Ethernet address followed by the statistics block, an array of 14 32-bit statistics counters. (See Appendix F). The driver ignores blk.

The keyboard monitor's Set Command lets the user change the Ethernet address of the controller. The names of the variables to set are: ADDR0, ADDR1, and ADDR2. ADDR0 must be an even number. An odd number makes it a multicast address. The address recognition routine compares the first word that arrives on the ether with ADDR0, the next word with ADDR1, and finally ADDR2.

The driver assumes that QE memory is assigned to bus addresses 100000-157776. RT-11 therefore resides in high QE memory. The System will load the device driver directly below the resident part of RT-11. The driver uses the packet buffers that fall into its data area. During initialization, the driver halts if it discovers that its data area does not fall within QE memory.

As supplied, the driver uses four packet buffers: one for transmit and three for receive. If this proves unsatisfactory, the allocation can be changed by reassembling the driver. Edit the source to change the values of NWBFRR and NWBFRX; these specify the number of buffers used for receive and transmit respectively.

# CHAPTER 8

# RSX-11M DRIVER

## 8.1. INTRODUCTION

The RSX-11M Ethernet driver described in this section supports the 3Com UE Controller which incorporates sixteen 2K-byte buffers residing in the UNIBUS address space. Data packets transmitted to/from the Ethernet go through these buffers as controlled by the driver. The number of transmit and receive packet buffers is configurable.

The UE Controller driver software follows standard practices for RSX drivers. The calling sequences, style, control and data structures are consistent with normal RSX conventions. Since multiple tasks use the Ethernet concurrently, the driver uses the type field in the Ethernet data link header to multiplex data. The driver does not introduce undue processing or storage overhead. In particular, packets are transferred directly from user memory into UE memory and vice-versa, without any extra buffering inside the kernel. The driver is simple, well commented, and may be used as a model for writing drivers under other operating system environments. Each routine has comments describing input/output parameters and the purpose of the routine. Manual intervention is never required under operational use.

## 8.2. QIO MACRO

The following section summarizes the QIO functions for the Ethernet driver. Table 8-1 lists the functions of the QIO macro that are valid for the Ethernet driver.

Table 8-1 - QIO Functions for Ethernet

| QIO Parameters | Function |
|---|---|
| IO.INL,...,<eaddr,size,ntb,nrb,mode> | Initialize. (Initialize Ethernet address, number of transmit and receive buffers, and address recognition filter mode.) |
| IO.INL!1,...,<eaddr,size,ntb,nrb,mode> | Initialize. (Initialize Ethernet address, number of transmit and receive buffers, address recognition filter mode, and enable Xerox NS Protocol |

| | |
|---|---|
| | Echo service.) |
| IO.STP,... | Stop the Ethernet controller. (Disables the Ethernet controller.) |
| IO.STP!1,... | Stop echo service. (Disables the Xerox NS Protocol Echo service portion of the driver but the Ethernet continues to operate.) |
| IO.CON,...,<type,rxefn> | Connect to Ethernet type. (Establishes ownership of Ethernet type supplied.) |
| IO.CAS,...,<type,rxast> | Connect to type with Asynchronous System Trap (AST). (Same as above and uses AST's to notify task of incoming packets.) |
| IO.DIS,...,<handle> | Disconnect from type. (Releases ownership of Ethernet type.) |
| IO.WLB,...,<bufadd,size,handle> | Write logical block. (Transmit buffer contents to Ethernet.) |
| IO.RLB,...,<bufadd,size,handle> | Read logical block. (Read Ethernet packet to user buffer.) |
| IO.RLB!1,...,<bufadd,size> | Return the Ethernet address of this controller (6 bytes). |
| IO.RLB!2,...,<bufadd,size> | Read statistics block (See Appendix D). |

Table 8-1 - Definitions

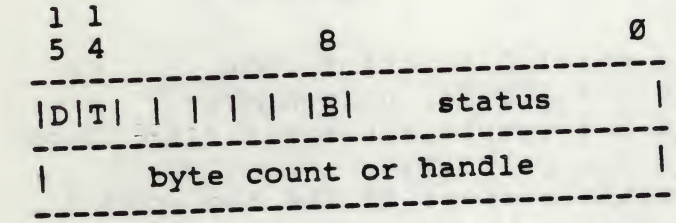
| | |
|---|---|
| eaddr | Starting address of a six-byte buffer containing the Ethernet address for this controller. |
| size | Data buffer size in bytes (must be even and greater than zero). |

| | |
|---|---|
| ntb | Number of transmit buffers to use. |
| nrb | Number of receive buffers to use. |
| mode | Filter mode for address recognition. |
| rxefn | Number of the event flag used to indicate the arrival of a receive packet for the associated Ethernet type. |
| rxast | Address of an asynchronous system trap routine to process received Ethernet packets of the associated type. |
| type | Sixteen bit Ethernet packet type. |
| bufadd | Starting address of the data buffer. |
| handle | Type access handle returned to the software from a successful connect function. |

## 8.3. STATUS RETURNS

The status and error conditions returned by the Ethernet driver in the I/O Status Block (IOSB) are shown in the diagram below. The first byte of the IOSB contains the status for the I/O function just performed. The second byte contains function dependent status bits. The second word of the IOSB contains the byte count transferred for either write or read functions, or the handle assigned to an Ethernet type following a successful connect function.

IOSB




IOSB Diagram Definitions:

| | |
|---|---|
| D (bit 15) | Error bit indicating that the maximum number of retries (as defined by the Ethernet) for transmission of a packet was exceeded. This is a fatal error and the Ethernet is reset and restarted. Any other I/O requests to the Ethernet may also fail. |
| T (bit 14) | Status bit indicating that the receive packet was truncated because the software byte count specified in the read was less than the number of bytes in the packet. |

B (bit 8) Non-zero indicates there are one or more receive buffers containing packets for this type. The software should issue a read request to free a buffer and continue doing so until this bit is returned as zero.

## 8.4. ETHERNET USAGE

Since the UE Controller uses buffers residing in UNIBUS address space, the RSX-11M system manager must be aware of possible conflict of memory use. Care should be exercised if the end of real memory exactly meets the beginning of UE buffer memory. RSX automatically extends its partitions into this memory space, thus causing the chance of spurious errors due to Ethernet operation. It is recommended that a hole in the memory addresses be provided to guarantee that RSX will not attempt to use Ethernet buffers as memory.

## 8.5. DETAILS ON ETHERNET DRIVER QIO FUNCTIONS

The Ethernet driver for RSX-11M performs six basic functions. These include initialize, stop, connect, disconnect, write (transmit) and read (receive). Optional to the above functions are initialize with echo service, stop echo service, connect with AST, read Ethernet address and read Ethernet statistics. Following are detailed descriptions of the QIO calls used with the Ethernet driver. The function dependent input parameters and error codes specific to each driver call are described. The general form of the QIO statement for RSX-11M is:

```
QIO$C   func,lun,efn,pri,isb,ast,prl
```

where

```
func   is the I/O function code
lun    is the logical unit number
efn    is the completion event flag number
pri    is the priority
isb    is the address of the two word I/O
         status block
ast    is the address of the AST service
         routine entry point
prl    is the function dependent parameter
         list <p1,...,p6>
```

### 8.5.1. IO.INL

IO.INL is the Ethernet driver initialization function. This initialization establishes the Ethernet address for the controller, the address recognition filtering mode and the number of buffers to use for transmit and receive. After the driver is loaded, this function must be issued to each controller to start its operation. The initialize QIO statement is as follows:

QIO$C IO.INL,...,<ethadd,size,ntxbuf,nrxbuf,mode>

A UE controller may be initialized to use either the address provided in the address ROM or any valid Ethernet address supplied by the caller. This address is defined in the ethadd parameter. The hardware address is used if the multicast bit (bit zero of the first word) is set in the ethadd buffer. The length of the ethadd buffer is given in the size parameter and must always have the value six. The next two parameters, ntxbuf and nrxbuf, define the number of transmit and receive buffers to be used by the Ethernet controller. The sum of these two values cannot exceed the number of buffers established in the driver configuration (file ENCFG.MAC) and cannot exceed sixteen. The mode parameter specifies the address filtering mode to be used by the controller and driver. The mode is a four-bit field which represents combinations of own address, multicast address, broadcast address and promiscuous mode. This four bit field is loaded directly into the UE receive CSR to enable the hardware address recognition. (On a QE controller, the driver emulates this address recognition scheme using the mode bits provided.)

An alternate form of the initialize function requires a modifier value of one (IO.INL!1) and performs the identical function with the addition that it enables the Xerox NS Echo server feature in the driver. The call is as follows:

QIO$C IO.INL!1,...,<ethadd,size,ntxbuf,nrxbuf,mode>

Once Xerox NS Protocol Echo Service is enabled, the driver echos any received Xerox NS Protocol Echo Packet to the originator if all of the following conditions are met:

1. The packet is addressed to this controller specifically.
2. The data link client type is 0600 hexadecimal and no user task is currently connected to this type.
3. The OIS client type is 2.
4. The echo type is 1.

The conditional assembly parameter ROSEBC in file ENCFG.MAC will, if defined, enable the echo service function to respond to broadcast packets.

Several error codes may be returned from the initialize function in the first byte of the I/O status block. These are:

| | |
|---|---|
| IE.ONL | initialization has already been performed |
| IE.RBG | buffer length error, or<br>buffer counts invalid, or<br>filter mode invalid |
| IE.BDV | invalid Ethernet address given (QE only) |

### 8.5.2. IO.STP

IO.STP disables the operation of the Ethernet for the given controller. The driver stops all activity, resets the controller and returns control to the caller. The controller may be restarted by issuing another initialize function. The stop command is issued in this manner:

```
QIO$C       IO.STP,...
```

A modifier of one to the stop function code (IO.STP!1) will disable the echo service function within the Ethernet driver but does not stop the rest of the driver functions from continuing to operate.

```
QIO$C       IO.STP!1,...
```

This stops only the echo service function within the driver. No errors are returned from the driver.

### 8.5.3. IO.CON

IO.CON is the general connect to Ethernet type function required before any task may perform any writes or reads to the Ethernet. A successful connect operation returns a 16-bit handle in the second I/O status block word. All subsequent QIO's pertaining to this type must provide this handle as an input parameter. The connected task is notified of received packets via the rxefn parameter - the driver sets this event flag whenever a packet is received for this type and clears the event flag when the last packet is read. The call is as follows:

```
QIO$C       IO.CON,...,<type,rxefn>
```

Error codes returned from the connect command include:

IE.DAA all type slots are used

### 8.5.4. IO.CAS

IO.CAS is similar to the above function except that it specifies an asynchronous system trap (AST) to process a received packet. It is called as follows:

```
QIO$C       IO.CAS,...,<type,rxast>
```

This function connects to the specified Ethernet type and establishes rxast as the entry point of the received buffer notification routine in the software task. On receipt of a packet of the specified type the driver issues an AST to the routine address provided to notify it of a new packet. The AST routine is executed asynchronously, similar to an interrupt routine. Due to the nature of the AST function within RSX, the AST routine is

best implemented a low level language such as MACRO-11. Upon entry to the AST routine, the software task is provided one word of information about the received packet - the handle. This value should be popped form the stack and used at a later time to read the packet data into a task buffer to free the Ethernet controller buffer. The AST program stack appears as follows:

| | |
|---|---|
| SP+10 | Event flag mask word |
| SP+06 | PS of task prior to AST |
| SP+04 | PC of task prior to AST |
| SP+02 | Task's directive status word |
| SP+00 | Handle for received packet type |

Once the AST routine has removed the handle from the stack it may return to the interrupted task via the ASTX system call. The error codes returned from the connect with AST are identical to those returned by the standard connect described above.

### 8.5.5. IO.WLB

IO.WLB performs a transmit of a single packet to the Ethernet. The packet data is provided by the calling task. It must include the Ethernet data link header (seven words) which should already include the destination Ethernet address. The Ethernet driver stores the source address and type field within the header before transmitting the packet. The transmit QIO takes this form:

```
QIO$C       IO.WLB,...,<bufadd,size,handle>
```

The packet data is provided in the first function dependent parameter bufadd. This includes the Ethernet header and any other data necessary up to the maximum packet size of 1514 bytes. The length of the packet is defined by the parameter size, which indicates the number of bytes to transmit, not including FCS. The type identification for this transmit is done by passing the handle returned by the connect function. This handle identifies the Ethernet type to the driver. On return to the calling task the packet will have been transmitted or an error message will be returned. Error codes returned from the transmit function are:

| | |
|---|---|
| IE.DNA | invalid handle |
| IE.RBG | buffer length out of range |
| IE.RSU | no transmit buffers available (try again) |
| IE.DNR | no transmit interrupt |
| IE.FHE | jam count exceeded |

### 8.5.6. IO.RLB

The Ethernet read logic has three functions. Depending on the modifier to the function code this either reads the next packet from the received complete list for the type, returns the Ethernet address currently used by this controller or returns

information from the statistics block kept by the driver. To read an Ethernet packet, the calling process first waits to be informed of the arrival of a packet with the specified type. The notification is done either by setting an event flag or by issuing an AST to a task routine as designated in the connect call (IO.CON or IO.CAS). The task then issues the packet read in this manner:

```
QIO$C       IO.RLB,...,<bufadd,size,handle>
```

The first two parameters specify the buffer address and number of bytes to read. The access handle is used by the driver to identify the Ethernet type and therefore determines which packet to transfer to bufadd. If no packets are waiting to be read for the designated type, an error is returned. The bufadd and size parameters, for ease of programming, should be set to the maximum Ethernet packet size, 1514 bytes, to account for a packet of any size. The actual packet size in bytes (not including FCS) is returned in the second word of the I/O status block on completion of the transfer.

The Ethernet address used by this controller may be read by issuing a read request with a modifier value of one (IO.RLB!1). The call is as follows:

```
QIO$C       IO.RLB!1,...,<bufadd,size>
```

The two parameters are the buffer specifier and length in bytes. The byte count should be six. The user task need not be connected to an Ethernet type to issue this call.

The Ethernet statistics block may be read by issuing a read function with a modifier value of two (IO.RLB!2). This returns up to 56 bytes of statistics data as described in Appendix D. These counters are updated by the driver and never zeroed. The call is as follows:

```
QIO$C       IO.RLB!2,...,<bufadd,size>
```

Error codes returned from the Ethernet driver read functions are described below:

```
IE.DNA  invalid handle
IE.RBG  buffer length out of range
IE.NBF  no receive packet pending
```

### 8.5.7. IO.DIS

IO.DIS disconnects the calling task from the Ethernet type associated with the given handle. This request cannot fail; the call is as follows:

```
QIO$C       IO.DIS,...,<handle>
```

If an invalid handle is given, this request is ignored.

## 8.6. CONFIGURING THE ETHERNET SOFTWARE

To properly configure the Ethernet driver for an RSX-11M installation, several questions must be answered. The results of these questions will be reflected in the Ethernet software generation by modifying the driver configuration file ENCFG.MAC. Once this file is properly modified, the Ethernet driver may then be assembled and task built with the files provided. At this point the 3Com controller boards may be installed into the backplane (if they are not already) and the driver may be loaded, initialized and used. The pertinent questions are:

1. Is this a Q-bus or UNIBUS system?

2. How many Ethernet controllers will be installed on this system? (The maximum value is four.)

3. What are their hardware addresses? On RSX-11M systems the buffer memory should be addressed as high as possible, so start at 700000 (or 600000 to use all sixteen buffers) and work down for each successive controller. Interrupt vectors can be chosen at any address which does not conflict with another device connected to the system.

4. How many units does each controller require? This value is determined by the number of tasks which require simultaneous access to data on the Ethernet. Note that a single task can connect to more than one type at once.

5. How many Ethernet types will be active? There is only one value for the entire driver, so the largest value over all the controllers should be used. The default is 8.

Once these questions are answered, ENCFG.MAC may be edited to reflect the system configuration. The answers to questions 2, 3, and 4 are reflected in the DVDEF statement in ENCFG.MAC. The DVDEF statement defines the number of controllers, the number of units on each controller, and the UNIBUS address of the CSR, the interrupt vector, the buffer memory address and number of buffers for each controller. A DVDEF statement looks like:

```
DVDEF       EN,cn,ua,va,nu,ba,op,nb
```

where:

```
cn      is the controller number, starting with zero.
ua      is the UNIBUS address of the CSR (octal).
va      is the interrupt vector address (octal).
nu      is the number of units on this controller.
ba      is the buffer address in 64 byte blocks
        (6000 represents 600000 octal).
op      (not used).
nb      is the number of buffers available.
```

Multiple controllers are represented by multiple DVDEF lines, incrementing the controller number (cn) for each successive one and specifying a unique ua, va and ba. The buffer count for each controller can be up to sixteen. The answers to the remaining questions are specified by assembly parameters defined at the end of the ENCFG.MAC file. The conditional parameter QE should be defined to specify a configuration using QE Ethernet boards. The maximum type count is defined by the symbol MAXTYP and may be changed to any value that suits the user. The memory overhead for each type is seven words per controller.

## 8.7. EXAMPLE

The following FORTRAN program example is provided to demonstrate the use of the Ethernet under RSX-11M.

```
      program enecho
c
c Sample FORTRAN program to demonstrate
c reception of Ethernet packets using
c the RSX driver.  This program transmits
c an echo packet over the Ethernet and
c waits for some echo server to return

c loop for the purpose of determining
c the total and average time elapsed
c in a single communications cycle with
c a constant host as a server.  The
c values printed by this routine represent
c the time including that used in QIO
c processing and driver packet handling.
c
      integer pkt(757),pktr(757),npkt
      integer iosb(2),parms(6),parmsr(6)
      integer handle,lun,ids,rxefn,retrys
      integer iorlb,iocon,iowlb,iodis
      logical wtpkt
c
      data npkt / 10000 /
      data rxefn / 7 /
```

```
c
c Define source packet data.
c
      data pkt(1),pkt(2),pkt(3) / -1,-1,-1 /
      data pkt(4),pkt(5),pkt(6) / 6,8,5 /      ! phoney source address.
      data pkt(7) / 6 /          ! client type.
      data pkt(10) / "1000 /
      data pkt(16) / "1000 /
      data pkt(23) / "400 /
      data lun / 1 /
c
c I/O function codes in the order of appearance.
c
      data iorlb / "1000 /
      data iocon / "15400 /
      data iowlb / "400 /
      data iodis / "16000 /
c
c Request the packet size.
c
  5     type 10
 10     format ('$Enter packet byte size:')
      accept 20, isz
 20     format (i5)
      if (isz.lt.60.or.isz.gt.1514) goto 5
c
c Assign the Ethernet access path.
c
      call asnlun(lun,'XX',0,ids)
      call chks(ids,1,6,'ASNLUN')
c
c Read the Ethernet address from the controller.
c
      call getadr(parms,pkt(4))
      parms(2)=6
      call wtqio(iorlb+1,lun,3,,iosb,parms,ids)
      call chks(ids,iosb,18,'read Ether address')
c
c Attach the Ethernet access path.
c
      parms(1)=6
      parms(2)=rxefn
      call wtqio(iocon,lun,3,,iosb,parms,ids)
      call chks(ids,iosb,10,'attach QIO')
      call clref(rxefn)
c
c Successful attach, save the handle.
c
      handle=iosb(2)
c
c Now send out an echo packet and wait for a reply.
c
      call getadr(parms,pkt)
      parms(2)=isz
```

```
      parms(3)=handle
      retrys=5
 25      call wtqio(iowlb,lun,5,,iosb,parms,ids)
      call chks(ids,iosb,8,'transmit')
      if (wtpkt(rxefn)) goto 30
c
c Process timeout.  Retry (at least a few times.)
c
      retrys=retrys-1
      if (retrys.gt.0) goto 25
      type 27
 27      format(' No server response, giving up.')
      stop
c
c Got a response.  Read the packet into our buffer.
c
 30      call getadr(parmsr,pktr)
      parmsr(2)=1514
      parmsr(3)=handle
      call wtqio(iorlb,lun,5,,iosb,parmsr,ids)
      call chks(ids,iosb,17,'read first packet')
c
c Set up our transmit packet with correct data.
c
      pkt(1)=pktr(4)
      pkt(2)=pktr(5)
      pkt(3)=pktr(6)
c
c Start the timer.
c
      time=secnds(0)
c
c Begin sending single packets to our server and
c waiting for each return packet.
c
      do 100 i = 1, npkt
      call wtqio(iowlb,lun,5,,iosb,parms,ids)
      call chks(ids,iosb,8,'transmit')
      if (wtpkt(rxefn)) goto 50
      type 40, i-1
 40      format(' Server disappeared after ',i5,' packets received.')
      stop
c
c Received the packet.  Read it.
c
 50      call wtqio(iorlb,lun,5,,iosb,parmsr,ids)
      call chks(ids,iosb,4,'read')
 100     continue
c
c Compute elapsed time and print results.
c
      time=secnds(time)
      type 22, npkt,time,float(isz)*npkt/time
 22      format(' Elapsed and average times, ',i5,' packets'/
```

```
     1 '  Total time:', f9.1, ' seconds'/
     2 '  Bytes/sec: ', f9.2)
c
c Disconnect the Ethernet access path.
c
      parms(1)=handle
      call wtqio(iodis,lun,5,,iosb,parms,ids)
      call chks(ids,iosb,10,'disconnect')
      end
c
c Subroutine to check the directive status
c and I/O status from a qio.  If an error
c is discovered, a message is printed on the
c controlling terminal and the job is aborted.
c
      integer function chks(ids,iosb,len,str)
c
c Establish correct types.
c
      integer len
      byte str(1),iosb,ids
      integer issuc
      data issuc / 1 /
c
c First check the directive status.
c
      if (ids.eq.issuc) goto 10
c
c Error in directive status.
c
      type 1, ids,(str(i),i=1,len),'.'
 1       format (' Directive error (',o3,') from ',20a1)
      stop
 10     continue
c
c Directive status ok, check I/O status.
c
      if (iosb.eq.issuc) goto 20
c
c Error in I/O status.
c
      type 2, iosb,(str(i),i=1,len),'.'
 2       format (' Status error (',o3,') from ',20a1)
      stop
 20     continue
c
c Done.
c
      return
      end
c
c Function subprogram to wait for a return packet
c from the server process.  A two-second timeout
c is incorporated into the wait logic to insure
```

```
c the program does not hang forever if there is
c no echo server around.
c
      logical function wtpkt(rxefn)
c
      integer isset,rxefn,timefn
c
      data isset / 2 /
      data timefn / 6 /
c
c Start the two second timer using event flag
c timefn.
c
      call mark(timefn,2,2,ids)
      call chks(ids,1,9,'mark time')
c
c Wait for either flag to appear.
c
      call wflor(timefn,rxefn)
c
c Clear them both, remembering which was set.
c
      call clref(rxefn,idsr)
      call clref(timefn,idst)
      call canmt(,ids)
c
c Check results.  If rxefn set, return the packet
c success code.  Else, return error status.
c
      wtpkt=.false.
      if(idst.ne.isset) wtpkt=.true.
      return
      end
```

# APPENDIX A

# ADDRESS RECOGNITION SOFTWARE

APPENDIX A - ADDRESS RECOGNITION SOFTWARE

```
        .title QEserv Q-bus/Ethernet Interrupt Service
        .list ttm
        .enable lc

a=%0 ;data pattern
b=%1 ;ether buffer number
c=%2 ;scratch
d=%3 ;scratch
t=%4 ;address
tt=%5       ;byte count
sp=%6
pc=%7

qercr= 160010  ;ether serial receiver control register
qexcr= 160012  ;transmitter control register

qetrp= 320     ;ether serial board trap location

qeaddr= 60000  ;origin of qe memory
bufcnt= 4000   ;number of bytes in a packet buffer
bigcnt= \d1514 ;largest legal size packet, no fcs
fcscnt= 4 ;number of bytes in fcs
rstart= bufcnt-bigcnt-fcscnt-2      ;buffer offset of rcvd packet

myaddr:   .word 0, 0, 0

pkttbl: .irp cnt, <0,1,2,3,4,5,6,7,10,11,12,13,14,15,16,17>
        .word cnt*bufcnt+qeaddr
        .endr

;various other cases
bcast:    halt ;broadcast packet
forme:    halt ;packet addressed to me
lngpkt:   halt ;packet is too long
fcserr:   halt ;frame check sequence error

;overhead for interrupt          ; 8.85 (time for IOT and BPT)
rcvbrk:   mov a, -(sp)           ; 3.56
        mov b, -(sp)          ; 3.56 subtotal 15.97

        mov qercr, b          ; 4.27
        bic #177760, b        ; 2.84
        mov b, a          ; 1.72
        asl b                 ; 1.72
        mov pkttbl(b), b      ; 4.27
        tst (b)               ; 3.14
        blt fcserr            ; 1.72
        beq lngpkt            ; 1.72
        add #rstart, b        ; 2.84
        bit #1, (b)           ; 4.26
        bne tbcast            ; 1.72 subtotal 30.22

        cmp (b)+, myaddr      ; 5.39
        bne ignpkt            ; 1.72
        cmp (b)+, myaddr+2    ; 5.39
        bne ignpkt            ; 1.72
        cmp (b)+, myaddr+4    ; 5.39
        beq forme         ; 1.72 subtotal 21.33

ignpkt:   bis #160, a             ; 1.84
        mov a, qercr          ; 4.68 subtotal  6.52

        mov (sp)+, b          ; 2.84
        mov (sp)+, a          ; 2.84
        rtt                ; 5.17 subtotal 10.85
;-------------------------------------------------------------
;                          grand total      84.89 microsecs
;
;overhead due to interrupt     26.82      32%
;overhead for packet           36.74      43%
;worst case compare time 21.33      25%

;best case compare time          7.11
;best case total time           70.57

;looking for broadcast instead of physical address takes less time
tbcast:   cmp (b)+, #177777    ; 4.26
        bne ignpkt            ; 1.72
        cmp (b)+, #177777     ; 4.26
        bne ignpkt            ; 1.72
        cmp (b)+, #177777     ; 4.26
        bne ignpkt            ; 1.72 subtotal 17.94

        br bcast          ; 1.72

        .end
```

# APPENDIX B

## TRANSMIT PACKET EXAMPLE

```
Oct 14 13:31 1981  example.mac Page 1


.title EXAMPLE  Examples of simple, busy waiting, QE programming

 a=%0    ;data pattern
b=%1    ;ether buffer number
c=%2    ;scratch
d=%3    ;scratch
t=%4    ;address
tt=%5   ;byte count
sp=%6
pc=%7

qercr= 164330   ;ether serial receiver control register
qexcr= 164332   ;transmitter control register

qebase= 100000  ;origin of QE memory in LSI-11 address space

;qexcr bits
xbn= 17         ;mask for buffer number
xwbn= 20        ;write buffer number
xclr= 40        ;clear transmit done
xinten= 100     ;transmit interupt enable
xdone= bytsgn   ;transmit done
jclr= 20000     ;clear jam
jinten= 40000   ;jam interupt enable
jam= snb        ;jam bit

;qercr bits
 bn= 17         ;mask for buffer number
.wbn= 20
rclr= 40
rinten= 100
r   = bytsgn

;useful combinations
xplete= jinten!xinten   ;transmit and jam interupt enables

bfsctl= 13      ;log number of bytes of in packet buffer
bufcnt= 4000    ;number of bytes in a packet buffer
bigcnt= ^d1514  ;largest legal size packet (add 4 more bytes for fcs)
smlcnt= ^d60    ;smallest legal size packet (add 4 more byte for fcs)
fcscnt= 4       ;number of bytes in fcs
rstart= bufcnt-bigcnt-fcscnt-2  ;buffer offset to first word of rcvd packet
ethlen= 2^      ;length of ethernet packet leader
nqebuf= 20      ;number of qe packet buffers in qe hardware
```

```
;transmits a packet, call with jsr pc, xpkt
 takes buffer index in b, size in tt, assumes data already in buffer
xpkt:   mov b, t                ;convert buffer number in b to offset in qe
        swab t                  ;shift left eight bits
        asl t
        asl t
        asl t                   ;origin of packet buffer in qe memory
        mov @bufcnt, d
        sub tt, d
        mov d, qebase(t)        ;offset of first word transmitted, into header
        bis @xwbn, b            ;set transmit bit
        mov b, @#qexcr
        bic @xwbn, b
        mov #17, -(sp)          ;initialize retry count
        mov #10000, d           ;generous timeout
xwait:  tstb @#qexcr            ;transmit done?
          bmi 1$
        tst @#qexcr             ;jam?
          bmi 4$
        sob d, xwait
        type <Transmit timed out >
        br 2$

;packet successfully transmitted, clear done
1$:     tst (sp)+               ;flush the retry count
        mov @#qexcr, d
        bic #^C<xbn, d
        cmp d, b
          beq 3$
          type <Transmitted out of order >
2$:       halt                  ;hardware malfunction
3$:     mov @xclr, @#qexcr      ;clear done
        rts pc

;come here on jam
4$:     dec (sp)
          bpl 5$
          halt                  ;something wrong with transmission subsystem

;exponential backoff
5$:     jsr pc, rangen          ;generate a random number
        mov d, -(sp)
        mov #15, d
        sub 2(sp), d            ;number of retries so far
        clr c
        cmp d, #12              ;clip at 10 (decimal retries)
          blos 6$
          mov #12, d
6$:     rol a                   ;the high order bits are much more random
        asl c                   ;exponentially increasing mean
        sob d, 6$
        mov (sp)+, d
        mov @#clr, @#qexcr      ;clear jam
        br xwait
```

jsr pc, stall

# APPENDIX C

## RECEIVE PACKET EXAMPLE

```
Oct 14 13:31 1981  example.mac Page 1


;takes buffer number in b and puts it into receive FIFO
 pkt:   bis #rwbn, b
        mov b, @#qercr
        bic #rwbn, b
rwaitn: mov #10000, d           ;generous timeout
rwait:  tstb @#qercr            ;received a packet?
          bmi 1$
        tst @#qercr             ;not functional, just wastes time
          bmi 3$                ;3.5C       3.50
3$:     sob d, rwait            ;           4.90
          type <Arrival timed out >
          halt                  ;hardware malfunction
          rts pc
1$:     mov @#qercr, -(sp)
        mov #rclr, @#qercr              ;clear done
        mov (sp)+, d            ;register before we cleared done
        bic #^Crbn, d
        cmp d, b
          beq 2$
          type <Arrived out of order >
          halt                  ;hardware malfunction
2$:     rts pc

;random number generator 4005*seed+33021
;Beware period of only 2^16; low order bits are not very random at all.
;See Knuth for all the appropriate warnings about random number generators.
 rngen: mov seed, a
        mov a, c
        asl c
        asl c
        add c, a
        swab c
        clrb c
        asl c
        add c, a
        add #33031, a
        mov a, seed
        rts pc

 seed: 0                ;seed for random number generator
        .end
```

# APPENDIX D

# ORDERING INFORMATION

## Ordering Information

| Model Number | Description |
|---|---|
| 3C300 | Q-bus Ethernet controller |
| 3C210-FA | Diagnostic on RX01-compatible diskette |
| 3C210-FB | Diagnostic on RX02-compatible diskette |
| 3C210-MA | Diagnostic on 800 bpi magnetic tape |
| 3C210-MB | Diagnostic on 1600 bpi magnetic tape |
| 3C211-FA | RT-11 driver and diagnostic on RX01-compatible diskette |
| 3C211-FB | RT-11 driver and diagnostic on RX02-compatible diskette |
| 3C211-MA | RT-11 driver and diagnostic on 800 bpi magnetic tape |
| 3C211-MB | RT-11 driver and diagnostic on 1600 bpi magnetic tape |
| 3C212-FA | RSX-11M driver and diagnostic on RX01-compatible diskette |
| 3C212-FB | RSX-11M driver and diagnostic on RX02-compatible diskette |
| 3C212-MA | RSX-11M driver and diagnostic on 800 bpi magnetic tape |
| 3C212-MB | RSX 11M driver and diagnostic on 1600 bpi magnetic tape |
| 3C220 | Reference Manual for Q-bus Ethernet controller |
| 3C800 | UNET Software program license |
| 3C810 | UNET Software binary copy |
| 3C100 | Ethernet transceiver |
| 3C110 | Transceiver cable 15 meter |

### Warranty

Any item which fails within 90 days of shipment will be repaired or replaced by 3Com free of charge. After 90 days a billable factory repair service is available.



**3Com** Corporation
computer communication corporation
1390 Shorebird Way
Mountain View, California 94043 USA
(415) 961-9602 Telex 345346

# APPENDIX B

## BIBLIOGRAPHY

1. DEC-INTEL-XEROX, "The Ethernet, A Local Computer Network, Data Link Layer and Physical Layer, Version 1.0", September 30, 1980.

2. Robert M. Metcalf and David R Boggs, "Ethernet: A Distributed Packet Switching for Local Computer Networks", CACM, 19:17, July 1976, pp. 395 - 404.

3. John F. Shoch and Jon A. Hupp, "Performance of an Ethernet Local Network - A Preliminary Report", Local Area Communications Network Symposium, Mitre and NBS, Boston, May 1979.

# APPENDIX F

## QE DRIVER STATISTICS

The following statistics are maintained by all QE drivers, and are accessable via a system call to the driver all values are kept as 32-bit integers. All values are initialized to zero and are never reset.

entxpc — transmit packet counter - incremented by one each time a packet is successfully transmitted.

entxcc — transmit collision counter - incremented ;by one each time a packet experiences a collision.

ennflc — network failure counter - incremented by one each time the number of collisions experienced by a single packet exceeds 15, indicating an Ethernet system failure. Subtracting 16 times this value from the entxcc value yields the number of collisions experienced during normal operation.

entxpl — transmit packet length - incremented by the size of the packet in bytes (including the 14 byte Ethernet header but excluding the preamble and FCS) each time a packet is successfully transmitted. The tpl and tpc fields

may be used to calculate an average transmit packet length.

enrxpc — receive packet counter - incremented by one each time a packet is received, regardless of its validity.

ensbfc — short bad FCS counter - incremented by one each time a packet is received that has an incorrect FCS and is shorter than the minimum size. Ensbfc + entxcc - 16* ennflc equals the number of collisions experienced on the Ethernet.

enlbfc — legal bad FCS counter - incremented by one each time a packet is received that has an incorrect FCS and is bigger than the minimum size. Ensbfc + enlbfc equals the total number of packets received by this station with incorrect FCS's.

ensgfc — short good FCS counter - incremented by one each time a packet is received that has a correct FCS but is longer than the maximum size.

enlgfc — long good FCS counter - incremented by one each time a packet is received that has a correct FCS but is shorter than the minimum size.

ennfmc not for me counter - incremented by one each time a packet is received that has a correct FCS and legal size but is destined for a specific Ethernet address other than our own.

enbroc broadcast counter - incremented by one each time a broadcast packet is received that has a correct FCS and legal size.

enmulc multicast counter - incremented by one each time a multicast packet is received that has a correct FCS and legal size.

enfmec for me counter - incremented by one each time a packet is received that has a correct FCS and legal size and is destined for this host, i.e., contains either our own address in the destination field, or a broadcast or multicast address.

enrxpl receive packet length - incremented by the size of the packet in bytes (including the 14 byte Ethernet header but excluding the preamble and FCS) each time a packet is received for this host. The enfmec and enrxpl counters may be used to calculate an average receive packet length.

**ensbfc + enlbfc + ensgfc + enlgfc + ennfmc + enfmec = enrxpc**

3Com Corporation
computer communication compatibility

## REQUEST FOR READERS COMMENTS

We want you to have documentation that meet your needs. Please use this form to participate directly in the documentation process and help us improve our publications.

Tell us of your experience regarding the usability, accuracy, readability, organization, and completeness of this publication only. If you have comments on the product this publication describes or if you wish to order publications, please contact your 3Com Representative directly.

Please fill out the form below and mail to 3Com. Thank you.

NAME_____ DATE_____

TITLE_____

COMPANY NAME/DEPARTMENT_____

ADDRESS_____

CITY_____STATE_____ZIP CODE_____

1. Please describe any errors you found in this document (include page number).

2. Does this document cover the information you expected or required? Please make suggestions for improvement.

3. Is this the correct level documentation for your needs? Any suggestions for other types of publications needed?

4. Did you have any difficulty understanding the material? Where?